

Constrained Fault-Tolerant Resource Allocation^{*}

Kewen Liao¹, Hong Shen^{1,2} and Longkun Guo³

¹School of Computer Science
The University of Adelaide, Adelaide, Australia

²School of Computer and Information Technology
Beijing Jiaotong University, Beijing, China

³School of Mathematics and Computer Science
Fuzhou University, Fuzhou, China

Abstract. We follow [18,25] and give a series of improved results for the Constrained Fault-Tolerant Resource Allocation (*FTRA*) problem. In *FTRA*, we are given a set of sites containing facilities as resources, and a set of clients consuming these resources. Specifically, each site i is allowed to open at most R_i facilities with cost f_i for each opened facility. Each client j requires an allocation of r_j facilities and connecting j to any facility at site i incurs a connection cost c_{ij} . The goal is to minimize the total cost of this resource allocation scenario.

FTRA generalizes the Unconstrained Fault-Tolerant Resource Allocation (*FTRA*_∞) [18] and the classical Fault-Tolerant Facility Location (*FTFL*) [13] problems: for every site i , *FTRA*_∞ does not have the constraint R_i , whereas *FTFL* sets $R_i = 1$. These problems are said to be uniform if all r_j 's are the same, and general otherwise.

In this paper, we strive to close the gap between *FTRA* and *FTFL* from the perspective of approximation. For the general metric *FTRA*, we first give an LP-rounding algorithm achieving the approximation ratio of 4. Then we show its reduction to *FTFL*, implying the ratio of 1.7245. For the uniform *FTRA*, we provide a 1.52-approximation primal-dual algorithm in $O(n^4)$, where n is the total number of sites and clients. For completeness in the line of [25] for *FTFL*, we also consider the Constrained Fault-Tolerant k -Resource Allocation (*KFTRA*) problem where additionally the total number of facilities can be opened across the sites is at most k . For the uniform *KFTRA*, we give the first constant-factor approximation algorithm with a factor of 4. Note that the above results carry over to *FTRA*_∞ and *KFTRA*_∞.

1 Introduction

In the *Constrained Fault-Tolerant Resource Allocation* (*FTRA*) problem introduced in [18], we are given a set \mathcal{F} of sites and a set \mathcal{C} of clients, where $|\mathcal{F}| = n_f$,

^{*} A small part of the Section 5 of this paper has appeared in [18].

$|\mathcal{C}| = n_c$ and $n = n_f + n_c$. Each site $i \in \mathcal{F}$ contains at most R_i ($R_i \geq 1$) facilities to open as resources and each client $j \in \mathcal{C}$ is required to be allocated r_j ($r_j \geq 1$) different facilities. Note that in *FTRA*, facilities at the same site are different and $\max_{j \in \mathcal{C}} r_j \leq \sum_{i \in \mathcal{F}} R_i$. Moreover, opening a facility at site i incurs a cost f_i and connecting j to any facility at i costs c_{ij} . The objective of the problem is to minimize the sum of facility opening and client connection costs under the resource constraint R_i . This problem is closely related to the *Unconstrained Fault-Tolerant Resource Allocation* ($FTRA_\infty$)¹ [18], the classical *Fault-Tolerant Facility Location* (*FTFL*) [13] and *Uncapacitated Facility Location* (*UFL*) [23] problems. Both $FTRA_\infty$ and *FTFL* are special cases of *FTRA*. In $FTRA_\infty$, R_i is not specified, whereas $\forall i \in \mathcal{F} : R_i = 1$ for *FTFL*. These problems are *uniform* if all r_j 's are same, otherwise they are *general*. If $\forall j \in \mathcal{C} : r_j = 1$, they all reduce to *UFL*. Fig. 1 displays an *FTRA* instance with a feasible solution. We notice that both *FTRA* and $FTRA_\infty$ have potential applications in today's distributed and high performance computing (DHPC) systems such as the cloud computing. The fault-tolerance attribute can be also viewed as the parallel processing capability of these systems. Unless elsewhere specified, we consider the problems in metric space, that is, the connection costs c_{ij} 's satisfy the metric properties like the triangle inequality and etc. Note that even the simplest non-metric *UFL* is hard to approximate better than $O(\log n)$ unless $NP \subseteq DTIME[n^{O(\log \log n)}]$ [24].

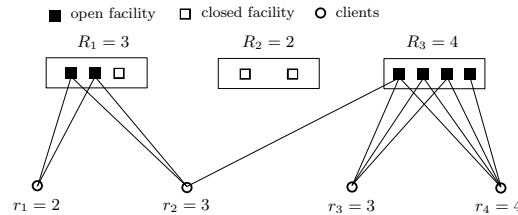


Fig. 1. An *FTRA* instance with a feasible solution

Related Work. Primal-dual and LP-rounding are two promising approaches in designing approximation algorithms for the facility location problems. Starting from the most basic and extensively studied *UFL* problem, there are JV [14], MMS [21] and JMS [12] primal-dual algorithms obtaining approximation ratios of 3, 1.861 and 1.61 respectively. In addition, Charikar and Guha [4] improved the result of the JV algorithm to 1.853 and Mahdian et al. [22] improved that of the JMS algorithm to 1.52, both using the standard cost scaling and greedy augmentation techniques. Shmoys et al. [23] first gave a filtering based LP-

¹ The problem is also called *Fault-Tolerant Facility Allocation* (*FTFA*) in [26] and *Fault-Tolerant Facility Placement* (*FTFP*) in [27]. Nevertheless, we reserve our name for identifying the different application-oriented resource allocation scenario. Our naming convention also follows from [6,10,15] for the set cover problems.

rounding algorithm achieving the constant ratio of 3.16. Following this, Guha and Khuller [7] improved the factor to 2.41 with greedy augmentation. Later, Chudak and Shmoys [5] came up with the clustered randomized rounding algorithm which further reduces the ratio to 1.736. Based on their algorithm, Sviridenko [24] applied pipage rounding to obtain 1.582-approximation. Byrka [1] achieved the ratio of 1.5 using a bi-factor result of the JMS algorithm. Recently, Li's more refined analysis in [17] obtained the current best ratio of 1.488, which is close to the 1.463 lower bound established by Guha and Khuller [7].

Comparing to UFL , $FTFL$ seems more difficult to approximate. For the general $FTFL$, the primal-dual algorithm in [13] yields a non-constant factor $O(\log n)$. Constant results exist only for the uniform case. In particular, Jain et al. [11,20] showed their MMS and JMS algorithms for UFL can be adapted to the uniform $FTFL$ while preserving the ratios of 1.861 and 1.61 respectively. Swamy and Shmoys [25] improved the result to 1.52. On the other hand, LP-rounding approaches are more successful for the general $FTFL$. Guha et al. [8] obtained the first constant factor algorithm with the ratio of 2.408. Later, this was improved to 2.076 by Swamy and Shmoys [25] with several rounding techniques. Recently, Byrka et al. [3] used dependent rounding and laminar clustering techniques to get the current best ratio of 1.7245.

$FTRA_\infty$ was first introduced by Xu and Shen [26] and they claimed a 1.861 approximation algorithm which runs in pseudo-polynomial time for the general case. Liao and Shen [18] studied the uniform case of the problem and presented a 1.52 approximation algorithm using a star-greedy approach. The general case of the problem was also studied by Yan and Chrobak [27] who gave a 3.16-approximation LP-rounding algorithm based on [23,5], and recently claimed the ratio of 1.575 [29] built on the work of [5,1,2,8]. They aim to close the approximation gap between $FTRA_\infty$ and UFL . On the other hand, due to the difficulties inherited from $FTFL$ and $FTRA_\infty$, it is still unknown what the approximation gap between $FTRA$ and $FTFL$ is.

In this paper, we strive to close this gap. However, there are several difficulties. First, despite the similar combinatorial structures of $FTRA_\infty$ and $FTRA$, the existing LP-rounding algorithms [27,29] for $FTRA_\infty$ can not be adapted for $FTRA$. The main reason is that the constraint R_i in $FTRA$ makes these algorithms produce infeasible solutions. In particular, the most recent work of [29] requires liberally splitting facilities and randomly opening them. This can not be done for both $FTRA$ and $FTFL$ as the splitting may cause more than R_i facilities to open, which is not a problem for $FTRA_\infty$. Second, in $FTFL$, $\max_{j \in C} r_j \leq n_f$, while r_j can be much larger than n_f in both $FTRA_\infty$ and $FTRA$. Therefore, the naive reduction idea of splitting the sites of an $FTRA$ instance and then restrict each site to have at most one facility will create an equivalent $FTFL$ instance with a possibly exponential size. Third, significantly more insights and heuristics are needed in addition to the previous work for solving $FTRA$ (both the general and the uniform cases) in polynomial time.

Our Contribution. For the general $FTRA$, we develop a *unified LP-rounding algorithm* through modifying the 4-approximation LP-rounding algorithm [25]

for *FTFL*. The unified algorithm can directly solve *FTRA*, $FTRA_\infty$ and *FTFL* with the same approximation ratio of 4. This is achieved by: 1) observing and proving some structured properties of the algorithm for directly rounding the optimal fractional solutions with values that might exceed one while ensuring the feasibility of the rounded solutions and the correctness of the algorithm; 2) exploiting the primal and dual complementary slackness conditions of the *FTRA* problem's LP formulation for achieving the ratio. Then we show *FTRA* reduces to *FTFL* using an *instance shrinking technique* inspired from the demand reduction trick of [29] for $FTRA_\infty$. It implies that these two problems share the same approximability in weakly polynomial time. Hence, from the *FTFL* result of [3], we obtain the ratio of 1.7245. For the non-metric *FTRA*, we get the first approximation factor of $O(\log^2 n)$ deduced from the results of [13,9].

For the uniform *FTRA*, we first present a naive *primal-dual algorithm* which runs in pseudo-polynomial time. In order to analyze it, we subsume dual fitting [11] and inverse dual fitting [26,19] analysis approaches into a simple and systematic *constraint-based analysis* to derive the ratio of 1.61. Later, with a *speed-up heuristic* applied to the primal-dual algorithm, we obtain the first strongly polynomial time algorithm for *FTRA* having the same ratio of 1.61 but with runtime $O(n^4)$. Moreover, with another similar heuristic applied to the greedy augmentation technique [8], the 3.16-approximation rounding result of [27] for the general $FTRA_\infty$ is improved to 2.408, and the previous 1.61 ratio for the uniform *FTRA* reduces to 1.52.

Lastly, we consider the Constrained Fault-Tolerant k -Resource Allocation (*KFTRA*) problem by adding an extra constraint that at most k facilities across the sites can be opened as resources. For the uniform *KFTRA*, we give a 4-approximation algorithm based on the work of [14,25]. In particular, the algorithm relies on a polynomial time *greedy pairing procedure* we develop for efficiently splitting sites into paired and unpaired facilities.

Note that the results shown in the paper directly hold for $FTRA_\infty$ and $KFTRA_\infty$, and the techniques developed will be useful for other variants of the resource allocation problems. For ease of analysis and implementation, the algorithms presented mostly follow the pseudocode style. Furthermore, we distinguish among pseudo-, weakly and strongly polynomial time algorithms w.r.t. the problem size n .

2 LP Basics and Properties

The *FTRA* problem has the following ILP formulation [18], in which solution variable y_i denotes the number of facilities to open at site i , and x_{ij} the number of connections between client j and site i .

$$\begin{aligned}
& \text{minimize} && \sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} c_{ij} x_{ij} \\
& \text{subject to} && \forall j \in \mathcal{C} : \sum_{i \in \mathcal{F}} x_{ij} \geq r_j \\
& && \forall i \in \mathcal{F}, j \in \mathcal{C} : y_i - x_{ij} \geq 0 \\
& && \forall i \in \mathcal{F} : y_i \leq R_i \\
& && \forall i \in \mathcal{F}, j \in \mathcal{C} : x_{ij}, y_i \in \mathbb{Z}^+
\end{aligned} \tag{1}$$

The problem's LP-relaxation (primal LP) and dual LP are the following:

$$\begin{aligned}
& \text{minimize} && \sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} c_{ij} x_{ij} \\
& \text{subject to} && \forall j \in \mathcal{C} : \sum_{i \in \mathcal{F}} x_{ij} \geq r_j \\
& && \forall i \in \mathcal{F}, j \in \mathcal{C} : y_i - x_{ij} \geq 0 \\
& && \forall i \in \mathcal{F} : y_i \leq R_i \\
& && \forall i \in \mathcal{F}, j \in \mathcal{C} : x_{ij}, y_i \geq 0
\end{aligned} \tag{2}$$

$$\begin{aligned}
& \text{maximize} && \sum_{j \in \mathcal{C}} r_j \alpha_j - \sum_{i \in \mathcal{F}} R_i z_i \\
& \text{subject to} && \forall i \in \mathcal{F} : \sum_{j \in \mathcal{C}} \beta_{ij} \leq f_i + z_i \\
& && \forall i \in \mathcal{F}, j \in \mathcal{C} : \alpha_j - \beta_{ij} \leq c_{ij} \\
& && \forall i \in \mathcal{F}, j \in \mathcal{C} : \alpha_j, \beta_{ij}, z_i \geq 0
\end{aligned} \tag{3}$$

From the formulation, it is easy to see that the problem becomes *FTFL* if $\forall i \in \mathcal{F} : R_i = 1$, and *FTRA*_∞ if the third resource constraint of the ILP is removed. Now if we let $(\mathbf{x}^*, \mathbf{y}^*)$ and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \mathbf{z}^*)$ be the optimal fractional primal and dual solutions, and $\text{cost}(\mathbf{x}, \mathbf{y})$ and $\text{cost}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z})$ be the cost functions (objective value functions) of any primal and dual solutions respectively. By the strong duality theorem, $\text{cost}(\mathbf{x}^*, \mathbf{y}^*) = \text{cost}(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \mathbf{z}^*)$. Moreover, the primal complementary slackness conditions (CSCs) are:

- (C1) If $x_{ij}^* > 0$ then $\alpha_j^* = \beta_{ij}^* + c_{ij}$.
- (C2) If $y_i^* > 0$ then $\sum_{j \in \mathcal{C}} \beta_{ij}^* = f_i + z_i^*$.

Dual CSCs are:

- (C3) If $\alpha_j^* > 0$ then $\sum_{i \in \mathcal{F}} x_{ij}^* = r_j$.
- (C4) If $\beta_{ij}^* > 0$ then $x_{ij}^* = y_i^*$.
- (C5) If $z_i^* > 0$ then $y_i^* = R_i$.

W.l.o.g., $(\mathbf{x}^*, \mathbf{y}^*)$ and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \mathbf{z}^*)$ have the following properties:

- (P1) $\forall j \in \mathcal{C} : \alpha_j^* > 0$ and $\sum_{i \in \mathcal{F}} x_{ij}^* = r_j$.
- (P2) $(\mathbf{x}^*, \mathbf{y}^*)$ is 'almost' complete, i.e. $\forall j \in \mathcal{C} : \text{if } x_{ij}^* > 0 \text{ then } x_{ij}^* = y_i^*$ (the complete condition) or there is at most one i s.t. $0 < x_{ij}^* < y_i^*$ where i is the farthest site connecting j . (cf. [5,25,29] for more details)

3 A Unified LP-Rounding Algorithm

The algorithm ULPR (Algorithm 1) starts by solving the primal and dual LPs to get the optimal solutions $(\mathbf{x}^*, \mathbf{y}^*)$ and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \mathbf{z}^*)$ to work with. In order to utilize the dual LP (LP (3)) for analyzing the approximation ratio of the output solution (\mathbf{x}, \mathbf{y}) , first of all, we encounter the difficulty (cf. Lemma 1) of bounding the $-\sum_{i \in \mathcal{F}} R_i z_i$ term in the dual objective function. To resolve this, we exploit the dual CSC (C5). The condition guides us to modify the 4-approximation algorithm of [25], and come up with Stage 1 of the algorithm ULPR which fully opens all (facilities of) sites with $y_i^* = R_i$ and put these sites into the set \mathcal{P} for pruning in future. Moreover, for successfully deriving the bound stated in Lemma 1, in the algorithm the client connections x_{ij}^* with the opened sites in \mathcal{P} are rounded up to $\lceil x_{ij}^* \rceil$; in the analysis the other primal and dual CSCs are also exploited. At the end of Stage 1, for each j , we calculate its established

connection \hat{r}_j , residual connection requirement \bar{r}_j and record its connected sites not in \mathcal{P} as set \mathcal{F}_j for the use of next stage.

Algorithm 1 ULPR: Unified LP-Rounding Algorithm

Input: $\mathcal{F}, \mathcal{C}, \mathbf{f}, \mathbf{c}, \mathbf{r}, \mathbf{R}$. **Output:** (\mathbf{x}, \mathbf{y})

Initialization: Solve LPs (2) and (3) to obtain the optimal fractional solutions $(\mathbf{x}^*, \mathbf{y}^*)$ and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \mathbf{z}^*)$. $\mathbf{x} \leftarrow \mathbf{0}, \mathbf{y} \leftarrow \mathbf{0}, \mathcal{P} \leftarrow \emptyset$

Stage 1: Pruning and Rounding

for $i \in \mathcal{F}$

if $y_i^* = R_i$ **do**

$y_i \leftarrow R_i$

$\mathcal{P} \leftarrow \mathcal{P} \cup \{i\}$

for $j \in \mathcal{C}$

if $x_{ij}^* > 0$ **do**

$x_{ij} \leftarrow \lceil x_{ij}^* \rceil$

set $\forall j \in \mathcal{C} : \hat{r}_j \leftarrow \sum_{i \in \mathcal{P}} x_{ij}, \bar{r}_j \leftarrow r_j - \hat{r}_j, \mathcal{F}_j \leftarrow \{i \in \mathcal{F} \setminus \mathcal{P} \mid x_{ij}^* > 0\}$

Stage 2: Clustered Rounding

set $\bar{\mathcal{C}} \leftarrow \{j \in \mathcal{C} \mid \bar{r}_j \geq 1\}$

while $\bar{\mathcal{C}} \neq \emptyset$

 /*2.1: Construct a cluster \mathcal{S} centered at j_o : directly use the S1 and S2 steps for *FTFL* (cf. page 6 of [25]) and tailor them for *FTRA**/*

$j_o \leftarrow \arg \min_j \{\alpha_j^* : j \in \bar{\mathcal{C}}\}$, order \mathcal{F}_{j_o} by non-decreasing site facility costs

 choose $\mathcal{S} \subseteq \mathcal{F}_{j_o}$ starting from the cheapest site in \mathcal{F}_{j_o} s.t. just $\sum_{i \in \mathcal{S}} y_i^* \geq \bar{r}_{j_o}$

if $\sum_{i \in \mathcal{S}} y_i^* > \bar{r}_{j_o}$ **do**

 split the last most expensive site $i_l \in \mathcal{S}$ into i_1 and i_2 : $y_{i_1}^* = \bar{r}_{j_o} - \sum_{i \in \mathcal{S} \setminus i_l} y_i^*$,

$y_{i_2}^* = y_{i_l}^* - y_{i_1}^*$; **forall** j : set $x_{i_1 j}^*, x_{i_2 j}^*$ s.t. $x_{i_1 j}^* + x_{i_2 j}^* = x_{i_l j}^*, x_{i_1 j}^* \leq y_{i_1}^*$

$x_{i_2 j}^* \leq y_{i_2}^*$ and update \mathcal{F}_j ; $\mathcal{S} \leftarrow \mathcal{S} \setminus \{i_l\} \cup \{i_1\}$ (now $\sum_{i \in \mathcal{S}} y_i^* = \bar{r}_{j_o}$)

 //2.2: Rounding around j_o and \mathcal{S}

 //2.2.1: Finish rounding \mathbf{y}

for $i \in \mathcal{S}$ //from the cheapest site

$y_i \leftarrow \lceil y_i^* \rceil$

$\bar{\mathcal{S}} \leftarrow \bar{\mathcal{S}} \cup \{i\}$ //maintain a set of already rounded sites

if $\sum_{i' \in \bar{\mathcal{S}}} y_{i'} \geq \bar{r}_{j_o}$

$y_i \leftarrow \bar{r}_{j_o} - \sum_{i' \in \bar{\mathcal{S}} \setminus i} y_{i'}$ (resetting y_i to make $\sum_{i' \in \bar{\mathcal{S}}} y_{i'} = \bar{r}_{j_o}$)

break

 //2.2.2: Finish rounding \mathbf{x}

for $j \in \bar{\mathcal{C}}$ //including j_o

if $\mathcal{F}_j \cap \mathcal{S} \neq \emptyset$

for $i \in \bar{\mathcal{S}}$ //order does not matter, could connect to the closest

$x_{ij} \leftarrow \min(\bar{r}_j, y_i)$

$\bar{r}_j \leftarrow \bar{r}_j - x_{ij}$

$\mathcal{F}_j = \mathcal{F}_j \setminus \mathcal{S}$

update $\bar{\mathcal{C}}$

The second major difficulty is that many LP-rounding algorithms round optimal solutions with values in $[0, 1]$, whereas for *FTRA*, our approach is unified

and general which directly rounds the solutions which might exceed 1 and later we shall analyze its correctness via establishing some useful properties. Stage 2 is inherited from the classical iterative clustering idea [5] for *UFL*. Similarly, our clustering and rounding terminate when all \bar{r}_j 's are satisfied, i.e. the set of not-fully-connected clients $\bar{\mathcal{C}} = \emptyset$ in the algorithm. Stage 2 consists of two substages 2.1 and 2.2, dealing with cluster construction and cluster guided rounding respectively. As pointed out in the algorithm, Stage 2.1 adopts the splitting idea of [25] for *FTFL* and tailors it for *FTRA*. In each iteration, it first picks the cluster center j_o with the smallest optimal dual value, and then builds a cluster \mathcal{S} around it which contains a subset of ordered sites in \mathcal{F}_{j_o} , starting from the cheapest site until $\sum_{i \in \mathcal{S}} y_i^* \geq r_{j_o}^-$. In order to maintain the invariant $\forall j \in \bar{\mathcal{C}} : \sum_{i \in \mathcal{F}_j} y_i^* \geq \bar{r}_j$ in every iteration, the stage then splits the last site $i_l \in \mathcal{S}$ into i_1 and i_2 , updates the client connections w.r.t. i_1 and i_2 , and in \mathcal{S} includes i_1 while excluding i_l to keep $\sum_{i \in \mathcal{S}} y_i^* = r_{j_o}^-$. Stage 2.2 does the final rounding steps around \mathcal{S} in addition to Stage 1 to produce a feasible integral solution (\mathbf{x}, \mathbf{y}) . This stage modifies and generalizes the rounding steps for *FTFL*. Its substage 2.2.1 rounds up the sites ($y_i^* \rightarrow \lceil y_i^* \rceil$) from the cheapest site in \mathcal{S} until $\bar{\mathcal{S}}$ (the set of sites rounded so far) just satisfies $\sum_{i' \in \bar{\mathcal{S}}} y_{i'} \geq r_{j_o}^-$ (now these $y_{i'}$'s are already integral). To make sure $\sum_{i' \in \bar{\mathcal{S}}} y_{i'} = r_{j_o}^-$ for bounding the site facility opening cost (cf. Lemma 2), the integral facility opening y_i of the last site i in $\bar{\mathcal{S}}$ is reset to $r_{j_o}^- - \sum_{i' \in \bar{\mathcal{S}} \setminus i} y_{i'}$, which is also integral. After the facilities at the sites in $\bar{\mathcal{S}}$ are opened according to the y_i 's constructed in stage 2.2.1, stage 2.2.2 then connects every client j in $\bar{\mathcal{C}}$ which has connections to the sites in \mathcal{S} (according to the x_{ij}^* 's) to $\min(\bar{r}_j, r_{j_o}^-)$ of these open facilities. It does this by iterating through all sites in $\bar{\mathcal{S}}$, setting x_{ij} 's and updating \bar{r}_j 's as described in the algorithm. At the end, for the run of next iteration, the sites in the cluster \mathcal{S} are excluded from \mathcal{F}_j , implying all clusters chosen in the iterations are disjoint; and $\bar{\mathcal{C}}$ is updated (at least j_o is removed from the set).

In the analysis, we first demonstrate the overall correctness of the algorithm ensured by the following properties. Note that some of the proofs in this section frequently refer to the content of Section 2.

(P3) After Stage 1, $\forall i \in \mathcal{P}, j \in \mathcal{C} : x_{ij} \leq R_i$ and $\bar{r}_j = r_j - \hat{r}_j \geq 0$.

Proof. The first part of the property is obvious since $x_{ij} = 0$ or $x_{ij} = \lceil x_{ij}^* \rceil \leq \lceil y_i^* \rceil \leq R_i$. For the second part, $\forall j \in \mathcal{C}$: if all x_{ij}^* are integers, we are done. Now we only need to consider j 's fractional x_{ij}^* with \mathcal{P} . By the previous property (P2), there is at most one fractional x_{ij}^* with \mathcal{P} because all y_i^* 's in \mathcal{P} are integers. Therefore, in Stage 1, at most one fractional x_{ij}^* is rounded up which will not make \hat{r}_j exceed r_j .

(P4) Stage 2.2.1 rounds $y_{i_1}^*$ (the optimal fractional opening of the last site i_1 in \mathcal{S} which is included in Stage 2.1) to at most $\lfloor y_{i_1}^* \rfloor$.

Proof. If $y_{i_1}^*$ is integral, the property clearly holds. Otherwise if $y_{i_1}^*$ is fractional, Case 1): if $\sum_{i' \in \bar{\mathcal{S}}} y_{i'} = r_{j_o}^-$ before resetting the last site i in Stage 2.2.1, this i definitely appears before i_1 in \mathcal{S} because otherwise $\sum_{i' \in \bar{\mathcal{S}}} \lceil y_{i'}^* \rceil$ will exceed

$r_{j_o}^-$, therefore i_1 is left unrounded; Case 2): If $\sum_{i' \in \bar{\mathcal{S}}} y_{i'} > r_{j_o}^-$, the last site i is possibly i_1 and if it is then $\bar{\mathcal{S}} = \mathcal{S}$, and from the algorithm we have rounded $y_{i_1} = r_{j_o}^- - \sum_{i' \in \mathcal{S} \setminus i_1} \lceil y_{i'}^* \rceil$ after resetting. If $y_{i_1} > \lfloor y_{i_1}^* \rfloor$ we get $\sum_{i' \in \mathcal{S} \setminus i_1} \lceil y_{i'}^* \rceil + \lfloor y_{i_1}^* \rfloor < r_{j_o}^-$ which is not possible since $\sum_{i \in \mathcal{S}} y_i^* = r_{j_o}^- = \sum_{i' \in \mathcal{S} \setminus i_1} y_{i'}^* + y_{i_1}^* = \left\lceil \sum_{i' \in \mathcal{S} \setminus i_1} y_{i'}^* \right\rceil + \lfloor y_{i_1}^* \rfloor \leq \sum_{i' \in \mathcal{S} \setminus i_1} \lceil y_{i'}^* \rceil + \lfloor y_{i_1}^* \rfloor$ (because $y_{i_1}^*$ is fractional). Hence, $y_{i_1}^*$ is rounded to at most $\lfloor y_{i_1}^* \rfloor$.

(P5) $\forall i \in \mathcal{F}$: given $0 < y_{i_1}^* + y_{i_2}^* = y_i^* \leq R_i$, then we have $\lfloor y_{i_1}^* \rfloor + \lceil y_{i_2}^* \rceil \leq \lceil y_i^* \rceil \leq R_i$.

Proof. We first have $\lfloor y_{i_1}^* \rfloor \leq y_{i_1}^*$ and $\lceil y_{i_2}^* \rceil < y_{i_2}^* + 1$, so $\lfloor y_{i_1}^* \rfloor + \lceil y_{i_2}^* \rceil < y_{i_1}^* + y_{i_2}^* + 1 = y_i^* + 1$. Now if y_i^* is integral, because $\lfloor y_{i_1}^* \rfloor + \lceil y_{i_2}^* \rceil$ is also integral, $\lfloor y_{i_1}^* \rfloor + \lceil y_{i_2}^* \rceil \leq \lceil y_i^* \rceil$. Otherwise if y_i^* is fractional, $\lfloor y_{i_1}^* \rfloor + \lceil y_{i_2}^* \rceil \leq \lfloor y_i^* \rfloor + 1 = \lceil y_i^* \rceil$. The property then follows from $\forall i \in \mathcal{F} : \lceil y_i^* \rceil \leq R_i$.

In summary, property (P3) shows the correctness of Stage 1 before going into Stage 2. (P4) and (P5) together ensure the splitting in Stage 2.1 and the rounding in Stage 2.2.1 produce feasible y_i 's for *FTRA*. This is because for any split sites i_1 and i_2 from i , (P4) guarantees at i_1 at most $\lfloor y_{i_1}^* \rfloor$ facilities are open, and (P5) makes sure that even $\lceil y_{i_2}^* \rceil$ facilities are opened at i_2 in the subsequent iterations of the algorithm, no more than R_i facilities in total actually get opened at i . Note that, (P5) also covers the situation that a site is repeatedly (recursively) split. Furthermore, in each iteration, Stage 2 at least fully connects the client j_o and considers all sites in the cluster \mathcal{S} centered at j_o . More importantly, the invariant $\forall j \in \bar{\mathcal{C}} : \sum_{i \in \mathcal{F}_j} y_i^* \geq \bar{r}_j$ is maintained for choosing the feasible cluster \mathcal{S} in Stage 2.1. This is true in the first iteration. In the subsequent iterations, the invariant still preserves because for any j with $\mathcal{F}_j \cap \mathcal{S} \neq \emptyset$ and is not fully connected in the current iteration, in the next iteration, $\sum_{i \in \mathcal{F}_j} y_i^*$ decreases by at most $r_{j_o}^-$ (because Stage 2.1 splits sites to maintain $\sum_{i \in \mathcal{S}} y_i^* = r_{j_o}^-$ and \mathcal{S} is excluded from \mathcal{F}_j in Stage 2.2.2) and \bar{r}_j decreases by exactly $r_{j_o}^-$ (from Stage 2.2.2). Therefore, the overall algorithm is correct.

Furthermore, the time complexity of the rounding stages of Algorithm 1 is $O(n^3)$ since each iteration of Stage 2 at least fully connects one of n_c clients which takes time $O(n^2)$. In the following, we separately bound the partial solution costs incurred in the stages involving rounding and then combine these costs for achieving the approximation ratio.

Lemma 1. *After pruning and rounding, the partial total cost from Stage 1 is $\sum_{j \in \mathcal{C}} \hat{r}_j \alpha_j^* - \sum_{i \in \mathcal{F}} R_i z_i^*$.*

Proof. $\forall i \in \mathcal{P}$:

$$\begin{aligned} \sum_{j \in \mathcal{C}} \lceil x_{ij}^* \rceil \alpha_j^* &= \sum_{j \in \mathcal{C}} \lceil x_{ij}^* \rceil c_{ij} + \sum_{j \in \mathcal{C}} \lceil x_{ij}^* \rceil \beta_{ij}^* \\ &= \sum_{j \in \mathcal{C}} \lceil x_{ij}^* \rceil c_{ij} + \sum_{j: x_{ij}^* = y_i^* = R_i} \lceil x_{ij}^* \rceil \beta_{ij}^* + \sum_{j: x_{ij}^* < y_i^* = R_i} \lceil x_{ij}^* \rceil \beta_{ij}^* \end{aligned}$$

$$\begin{aligned}
&= \sum_{j \in \mathcal{C}} \lceil x_{ij}^* \rceil c_{ij} + \sum_{j: x_{ij}^* = y_i^* = R_i} \lceil x_{ij}^* \rceil \beta_{ij}^* \\
&= \sum_{j \in \mathcal{C}} \lceil x_{ij}^* \rceil c_{ij} + R_i \left(\sum_{j: x_{ij}^* = y_i^* = R_i} \beta_{ij}^* + \sum_{j: x_{ij}^* < y_i^* = R_i} \beta_{ij}^* \right) \\
&= \sum_{j \in \mathcal{C}} \lceil x_{ij}^* \rceil c_{ij} + R_i \sum_{j \in \mathcal{C}} \beta_{ij}^* \\
&= \sum_{j \in \mathcal{C}} \lceil x_{ij}^* \rceil c_{ij} + R_i f_i + R_i z_i^*
\end{aligned}$$

The first equality is due to the condition (C1), the third, fourth and fifth is because by (C4) we have $\forall i \in \mathcal{P}, j \in \mathcal{C}$: if $\beta_{ij}^* > 0$ then $x_{ij}^* = y_i^* = R_i$, so $x_{ij}^* < y_i^* = R_i$ implies $\beta_{ij}^* = 0$ (by the contraposition in logic) and also $\sum_{j: x_{ij}^* < y_i^* = R_i} \beta_{ij}^* = 0$. The last equality is obtained from (C2), and the fact that $\forall i \in \mathcal{P}: y_i^* = R_i > 0$.

Summing both sides over all $i \in \mathcal{P}$, we can then bound the cost of Stage 1:

$$\begin{aligned}
\sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{C}} \lceil x_{ij}^* \rceil c_{ij} + \sum_{i \in \mathcal{P}} R_i f_i &= \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{C}} \lceil x_{ij}^* \rceil \alpha_j^* - \sum_{i \in \mathcal{P}} R_i z_i^* \\
&= \sum_{j \in \mathcal{C}} \hat{r}_j \alpha_j^* - \sum_{i \in \mathcal{F}} R_i z_i^*.
\end{aligned}$$

The second equality follows from the Stage 1 $\forall j \in \mathcal{C}: \hat{r}_j = \sum_{i \in \mathcal{P}} \lceil x_{ij}^* \rceil$, and the condition (C5): if $z_i^* > 0$ then $y_i^* = R_i$, so $y_i^* < R_i$ implies $z_i^* = 0$.

Lemma 2. *After finishing rounding \mathbf{y} , the partial site facility opening cost from Stage 2.2.1 is at most $\sum_{i \in \mathcal{F} \setminus \mathcal{P}} f_i y_i^*$.*

Proof. Facilities at sites $i \in S \subseteq \mathcal{F} \setminus \mathcal{P}$ are opened in f_i 's non-decreasing order in Stage 2.2.1: In any iteration of the algorithm with picked cluster \mathcal{S} , before rounding we have $\sum_{i \in \mathcal{S}} y_i^* = r_{j_o}^-$; after rounding set $\bar{\mathcal{S}}$ is formed starting from the cheapest site in \mathcal{S} s.t. $\sum_{i' \in \bar{\mathcal{S}}} y_{i'} = r_{j_o}^-$. This makes the opening cost of all sites in cluster $\bar{\mathcal{S}}$ at most $\sum_{i \in \mathcal{S}} f_i y_i^*$. The lemma then follows from the fact that all chosen clusters are disjoint in the algorithm.

Lemma 3. *After finishing rounding \mathbf{x} , the partial connection cost from Stage 2.2.2 is at most $3 \sum_{j \in \mathcal{C}} \bar{r}_j \alpha_j^*$.*

Proof. Let site i lie in the cluster \mathcal{S} centered at j_o . If j is already connected to i ($x_{ij}^* > 0$), then $c_{ij} \leq \alpha_j^*$ from the condition (C1). Otherwise, if j connects to i after rounding, from the algorithm, it implies $\alpha_{j_o}^* \leq \alpha_j^*$ (because j with the smallest α_j^* is always chosen as j_o) and $\mathcal{F}_j \cap \mathcal{S} \neq \emptyset$. Fig. 2 then displays the case $\mathcal{F}_j \cap \mathcal{S} \neq \emptyset$ where initially j connects to i' and it is connected to i after rounding.

By the triangle inequality, we have $c_{ij} \leq c_{i'j} + c_{ij_o} + c_{i'j_o}$. Also, it is true that $x_{i'j}^*, x_{ij_o}^*, x_{i'j_o}^* > 0$, so from (C1) we have $c_{i'j} \leq \alpha_j^*$ and $c_{ij_o}, c_{i'j_o} \leq \alpha_{j_o}^*$. Hence, $c_{ij} \leq \alpha_j^* + 2\alpha_{j_o}^* \leq 3\alpha_j^*$. Summing up both sides of this connection cost bound, we have $\sum_{i \in \mathcal{F} \setminus \mathcal{P}} \sum_{j \in \mathcal{C}} c_{ij} \leq 3 \sum_{j \in \mathcal{C}} \bar{r}_j \alpha_j^*$ and the lemma follows. Note that Fig. 2 does not show multiplicity of the connection between any client and site in an *FTRA* solution. It is merely for simplicity and will not affect the correctness of the proof.

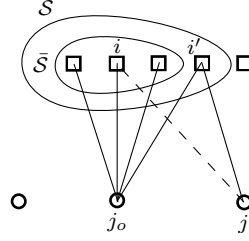


Fig. 2. Illustration of bounding the connection costs

Theorem 1. *Algorithm ULPR is 4-approximation for FTRA.*

Proof. Adding up the partial cost bounds stated in the previous lemmas, the total cost $\text{cost}(\mathbf{x}, \mathbf{y})$ is therefore at most $\sum_{j \in \mathcal{C}} \hat{r}_j \alpha_j^* - \sum_{i \in \mathcal{F}} R_i z_i^* + \sum_{i \in \mathcal{F} \setminus \mathcal{P}} f_i y_i^* + 3 \sum_{j \in \mathcal{C}} \bar{r}_j \alpha_j^*$. Also, we have $\text{cost}(\mathbf{x}^*, \mathbf{y}^*) = \sum_{i \in \mathcal{F}} f_i y_i^* + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} c_{ij} x_{ij}^* = \text{cost}(\alpha^*, \beta^*, \mathbf{z}^*) = \sum_{j \in \mathcal{C}} r_j \alpha_j^* - \sum_{i \in \mathcal{F}} R_i z_i^* = \sum_{j \in \mathcal{C}} \hat{r}_j \alpha_j^* + \sum_{j \in \mathcal{C}} \bar{r}_j \alpha_j^* - \sum_{i \in \mathcal{F}} R_i z_i^*$, so $\text{cost}(\mathbf{x}, \mathbf{y}) \leq \sum_{j \in \mathcal{C}} \hat{r}_j \alpha_j^* - \sum_{i \in \mathcal{F}} R_i z_i^* + \left(\sum_{j \in \mathcal{C}} \hat{r}_j \alpha_j^* + \sum_{j \in \mathcal{C}} \bar{r}_j \alpha_j^* - \sum_{i \in \mathcal{F}} R_i z_i^* \right) + 3 \sum_{j \in \mathcal{C}} \bar{r}_j \alpha_j^* = 4 \sum_{j \in \mathcal{C}} \bar{r}_j \alpha_j^* + 2 \sum_{j \in \mathcal{C}} \hat{r}_j \alpha_j^* - 2 \sum_{i \in \mathcal{F}} R_i z_i^* = 4 \text{cost}(\alpha^*, \beta^*, \mathbf{z}^*) - 2 \left(\sum_{j \in \mathcal{C}} \hat{r}_j \alpha_j^* - \sum_{i \in \mathcal{F}} R_i z_i^* \right) \leq 4 \text{cost}(\mathbf{x}^*, \mathbf{y}^*)$. The last inequality follows from the fact that $\left(\sum_{j \in \mathcal{C}} \hat{r}_j \alpha_j^* - \sum_{i \in \mathcal{F}} R_i z_i^* \right)$ is the cost of Stage 1 (cf. Lemma 1) which is nonnegative.

4 Reduction to FTFL

Recently, the authors in [29] proposed a demand reduction trick that reduces any FTRA_∞ instance with arbitrarily large r_j to another small FTRA_∞ instance with polynomially bounded r_j . The direct consequence of this is that FTRA_∞ is then reducible to *FTFL*, since we are able to naively split the sites of the small FTRA_∞ instance and the resulting instance is equivalent to an *FTFL* instance with a polynomial size. Because *FTRA* and FTRA_∞ have similar combinatorial structures, the question then becomes whether *FTRA* reduces to *FTFL* as well. In the following, we give an affirmative answer to this with the instance shrinking technique.

Compared to the reduction in [29] for $FTRA_\infty$, first, the instance shrinking technique for $FTRA$ does not rely on the complete assumption of the optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ of LP (2). Recall the property (P2) that $(\mathbf{x}^*, \mathbf{y}^*)$ is complete when $\forall j \in \mathcal{C} : \text{if } x_{ij}^* > 0 \text{ then } x_{ij}^* = y_i^*$. The reason is that in $FTRA$, the number of facilities at site i is bounded by R_i . This will cause inconvenience in splitting sites to construct the complete solution and then continuing the reduction. So rather we follow the similar settings in [28] for $FTRA_\infty$, which does not require $(\mathbf{x}^*, \mathbf{y}^*)$ to be complete. Second, the technique is more general than the demand reduction. This is because the purpose of the technique changes to reducing any $FTRA$ instance with arbitrarily large R_i to another small $FTRA$ instance with polynomially bounded R_i^s , which works for $FTRA_\infty$ as well since an $FTRA_\infty$ instance can be treated as an $FTRA$ instance with all R_i 's set to be $\max_{j \in \mathcal{C}} r_j$. The small $FTRA$ instance is then equivalent to an $FTFL$ instance with a polynomial size $(\sum_{i \in \mathcal{F}} R_i^s)$, implying $FTRA$ and $FTFL$ share the same approximability in weakly polynomial time. Third, in order to bound R_i^s polynomially, the technique heavily relies on the following *subtle* observation for carrying out instance shrinking.

Claim. $(\mathbf{x}^*, \mathbf{y}^*)$ remains to be the optimal solution even if R_i is replaced with $\lceil y_i^* \rceil$ in LP (2).

Proof. Denote the instance with parameter R_i as \mathcal{I}_o , and \mathcal{I} after replacing R_i with $\lceil y_i^* \rceil$. On one hand, solving \mathcal{I} will not yield any better optimal solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ with $\text{cost}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) < \text{cost}(\mathbf{x}^*, \mathbf{y}^*)$, because this $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ is also feasible to \mathcal{I}_o , which contradicts the optimality of $(\mathbf{x}^*, \mathbf{y}^*)$ for \mathcal{I}_o . On the other hand, $\text{cost}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) > \text{cost}(\mathbf{x}^*, \mathbf{y}^*)$ is not possible since $(\mathbf{x}^*, \mathbf{y}^*)$ is also a feasible solution to \mathcal{I} as $y_i^* \leq \lceil y_i^* \rceil$, which contradicts the optimality of $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ for \mathcal{I} . Hence, $(\mathbf{x}^*, \mathbf{y}^*)$ stays optimal for \mathcal{I} .

Now in the reduction phase of the instance shrinking technique, initially we can consider the equivalent $FTRA$ instance \mathcal{I} with $\forall i \in \mathcal{F} : R_i = \lceil y_i^* \rceil$ and the same optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$. Then, $(\mathbf{x}^*, \mathbf{y}^*)$ is split into a large integral solution with $y_i^l = \max(0, \lfloor y_i^* \rfloor - 1)$ and $x_{ij}^l = \min(\lfloor x_{ij}^* \rfloor, y_i^l)$, and a small fractional solution with $y_i^s = y_i^* - y_i^l$ and $x_{ij}^s = x_{ij}^* - x_{ij}^l$, for all $i \in \mathcal{F}, j \in \mathcal{C}$. Let the tuple $\langle \mathcal{F}, \mathcal{C}, \mathbf{f}, \mathbf{c}, \mathbf{r}, \mathbf{R} \rangle$ represent the instance \mathcal{I} , the reduction then proceeds by splitting \mathcal{I} into a large instance $\mathcal{I}^l : \langle \mathcal{F}, \mathcal{C}, \mathbf{f}, \mathbf{c}, \mathbf{r}^l, \mathbf{R}^l \rangle$ and a small instance $\mathcal{I}^s : \langle \mathcal{F}, \mathcal{C}, \mathbf{f}, \mathbf{c}, \mathbf{r}^s, \mathbf{R}^s \rangle$ according to $(\mathbf{x}^l, \mathbf{y}^l)$ and $(\mathbf{x}^s, \mathbf{y}^s)$. In particular, these two instances differ at two parameters \mathbf{r} and \mathbf{R} , where we let $r_j^l = \sum_{i \in \mathcal{F}} x_{ij}^l$, $r_j^s = r_j - r_j^l$ and $R_i^l = y_i^l$, $R_i^s = \lceil y_i^* \rceil - y_i^l$. Note that although the above splitting idea of the technique is inspired from the demand reduction for $FTRA_\infty$, the focus on splitting R_i is essentially different from reducing r_j . Also, here we can see that the construction of the shrunk instance \mathcal{I} with $R_i = \lceil y_i^* \rceil$ is crucial for bounding R_i^s , since if the original R_i is used, R_i^s can not be bounded and the technique will not work. In the following, the first lemma is mostly from the original splitting idea where we provide a simpler proof for it. The second is

directly from our instance shrinking and splitting on R_i . As shown later in the proof of Theorem 2, these lemmas are necessary for the approximation preserving reduction from \mathcal{I} to \mathcal{I}^s .

Lemma 4. $(\mathbf{x}^l, \mathbf{y}^l)$ is a feasible integral solution to \mathcal{I}^l and $(\mathbf{x}^s, \mathbf{y}^s)$ is a feasible fractional solution to \mathcal{I}^s .

Proof. According to the LP (2), it is trivial to see the feasibility of the integral solution $(\mathbf{x}^l, \mathbf{y}^l)$. For the fractional solution $(\mathbf{x}^s, \mathbf{y}^s)$, since $r_j = \sum_{i \in \mathcal{F}} x_{ij}^*$, $r_j^l = \sum_{i \in \mathcal{F}} x_{ij}^l$, $r_j^s = r_j - r_j^l$ and $x_{ij}^s = x_{ij}^* - x_{ij}^l$, we have $r_j^s = \sum_{i \in \mathcal{F}} x_{ij}^s$ and the first constraint of the LP holds. Further, it is easy to see $y_i^s \leq R_i^s$ and we are left to show the second constraint $\forall i \in \mathcal{F}, j \in \mathcal{C} : y_i^s - x_{ij}^s \geq 0$ holds, i.e. $y_i^* - y_i^l \geq x_{ij}^* - \min(\lfloor x_{ij}^* \rfloor, y_i^l)$. Consider two cases: 1) $y_i^l \leq \lfloor x_{ij}^* \rfloor$, then the inequality obviously follows from $y_i^* \geq x_{ij}^*$; 2) $y_i^l > \lfloor x_{ij}^* \rfloor$, the inequality $rhs = x_{ij}^* - \lfloor x_{ij}^* \rfloor$, and $lhs = y_i^* - \max(0, \lfloor y_i^* \rfloor - 1)$ after substituting y_i^l . Now again consider two sub cases: 2.1) $\lfloor y_i^* \rfloor \geq 1$, then $lhs \geq 1$ while $rhs \leq 1$, so $lhs \geq rhs$ and the inequality follows; 2.2) $\lfloor y_i^* \rfloor < 1$, then $lhs = y_i^*$, and since $1 > y_i^* \geq x_{ij}^*$, $\lfloor x_{ij}^* \rfloor = 0$ and $rhs = x_{ij}^*$, then the inequality follows. Overall, (x_{ij}^s, y_i^s) is a feasible solution.

Lemma 5. For the instances \mathcal{I}^l and \mathcal{I}^s the following holds:

- (i) $\max_{j \in \mathcal{C}} r_j^l \leq \sum_{i \in \mathcal{F}} R_i^l$ and $\max_{j \in \mathcal{C}} r_j^s \leq \sum_{i \in \mathcal{F}} R_i^s$.
- (ii) $R_i^s \in \{0, 1, 2\}$.

Proof. (i) The previous lemma and the constraints of the LP (2) together ensure the bounds that $\forall j \in \mathcal{C} : r_j^l \leq \sum_{i \in \mathcal{F}} x_{ij}^l \leq \sum_{i \in \mathcal{F}} y_i^l \leq \sum_{i \in \mathcal{F}} R_i^l$ and $r_j^s \leq \sum_{i \in \mathcal{F}} x_{ij}^s \leq \sum_{i \in \mathcal{F}} y_i^s \leq \sum_{i \in \mathcal{F}} R_i^s$.

(ii) We have $R_i^s = \lceil y_i^* \rceil - \max(0, \lfloor y_i^* \rfloor - 1)$ after the substitution of y_i^l . If $\lfloor y_i^* \rfloor \geq 1$, then $R_i^s = \lceil y_i^* \rceil - \lfloor y_i^* \rfloor + 1 \in \{1, 2\}$, otherwise if $\lfloor y_i^* \rfloor < 1$, then $R_i^s = \lceil y_i^* \rceil \in \{0, 1\}$.

Theorem 2. If there is a ρ -approximation polynomial-time algorithm \mathcal{A} for $FTRA$ with polynomially bounded R_i , then there is also a polynomial-time ρ -approximation algorithm \mathcal{A}' for $FTRA$.

Proof. We will describe such an algorithm \mathcal{A}' . It first does the instance shrinking and splitting as described before for any instance \mathcal{I} of $FTRA$. From (i) of Lemma 5, the split instances \mathcal{I}^l and \mathcal{I}^s are valid. From (ii), \mathcal{I}^s has polynomially bounded R_i^s . Note that if $R_i^s = 0$, we can safely remove this site i in \mathcal{I}^s , and set the solution $\forall j \in \mathcal{C} : x_{ij}^s, y_i^s = 0$ when later combining it with \mathcal{I}^l . Then, \mathcal{A}' uses \mathcal{A} as a subroutine to solve \mathcal{I}^s to obtain a feasible integral solution $(\bar{x}_{ij}^s, \bar{y}_i^s)$. From Lemma 4, $(\mathbf{x}^s, \mathbf{y}^s)$ is feasible to \mathcal{I}^s so $cost(\bar{\mathbf{x}}^s, \bar{\mathbf{y}}^s) \leq \rho \cdot cost(\mathbf{x}^s, \mathbf{y}^s)$. Finally, \mathcal{A}' combines $(\bar{x}_{ij}^s, \bar{y}_i^s)$ with the readily available constructed integer solution (x_{ij}^l, y_i^l) for \mathcal{I}^l . Because (x_{ij}^l, y_i^l) is a feasible integral solution to \mathcal{I}^l , then when combined, they form a feasible integral solution to \mathcal{I} as $r_j^l + r_j^s = r_j$ and $R_i^l + R_i^s = R_i = \lceil y_i^* \rceil$. The only thing left is to prove the combined solution from

\mathcal{A}' is ρ -approximation, i.e., $\text{cost}(\mathbf{x}^l, \mathbf{y}^l) + \text{cost}(\bar{\mathbf{x}}^s, \bar{\mathbf{y}}^s) \leq \rho \cdot \text{cost}(\mathbf{x}^*, \mathbf{y}^*)$. This follows from $\text{cost}(\mathbf{x}^l, \mathbf{y}^l) + \text{cost}(\bar{\mathbf{x}}^s, \bar{\mathbf{y}}^s) \leq \text{cost}(\mathbf{x}^l, \mathbf{y}^l) + \rho \cdot \text{cost}(\mathbf{x}^s, \mathbf{y}^s) \leq \rho \cdot (\text{cost}(\mathbf{x}^l, \mathbf{y}^l) + \text{cost}(\mathbf{x}^s, \mathbf{y}^s))$ and $\mathbf{x}^l + \mathbf{x}^s = \mathbf{x}^*, \mathbf{y}^l + \mathbf{y}^s = \mathbf{y}^*$.

Corollary 1. *FTRA is reducible to FTFL in weakly polynomial time.*

Proof. Any instance of *FTRA* with polynomially bounded R_i can be treated as an equivalent *FTFL* instance with the facility size $\sum_{i \in \mathcal{F}} R_i$ which is also polynomial. Then any polynomial time algorithm solves *FTFL* with ratio ρ can become the algorithm \mathcal{A} for \mathcal{A}' in the previous theorem to solve *FTRA* with the same ratio. In addition, the reduction requires solving the LP first to obtain $(\mathbf{x}^*, \mathbf{y}^*)$ which takes weakly polynomial time.

Therefore, from the above corollary and the result of [3] for the metric *FTFL*, we get the ratio of 1.7245 for the metric *FTRA*. Also, from the results of [13,9], we can deduce that the non-metric *FTFL* has the approximation ratio of $O(\log^2 n)$ in strongly polynomial time. This is because Jain and Vazirani [13] proved that *FTFL* reduces to *UFL* with a ratio loss of $O(\log n)$, and Hochbaum [9] showed that the non-metric *UFL* can be approximated with the ratio of $O(\log n)$. For the non-metric *FTRA*, we can achieve the same ratio in weakly polynomial time due to the reduction to *FTFL* first. Moreover, in future, any improved ratio for *FTFL* will directly hold for *FTRA*.

5 The Uniform *FTRA*

The reduction results in the previous section also imply that the uniform *FTRA* reduces to the uniform *FTFL*, so it achieves the ratio of 1.52 as in [25], but in weakly polynomial time. In this section, we show the same ratio can be obtained in strongly polynomial time without using the instance shrinking technique, but with a primal-dual algorithm and two speed-up heuristics. Note that the algorithms in the following work for the general *FTRA* as well. The uniform condition is only necessary in the analysis (Lemma 13).

We begin with a naive primal-dual (PD) algorithm (Algorithm 2) for *FTRA* with an approximation ratio of 1.61 and then present the first speed-up heuristic to improve the complexity of the algorithm to strongly polynomial $O(n^4)$. W.l.o.g., the PD algorithm assumes that each client j makes r_j connections and each connection is associated with a *port* of j denoted by $j^{(q)}$ ($1 \leq q \leq r_j$). Also, the function $\phi(j^{(q)})$ represents the facility/site a client j 's q -th port is connected with and the variable p_j keeps track of the port of the client j to be connected. The algorithm then gradually connects clients in the port order from 1 to r_j , as well as increasing the solution (\mathbf{x}, \mathbf{y}) from $(\mathbf{0}, \mathbf{0})$ in its *actions* in response to some *events* controlled by a global time t that increases monotonically from 0. All events *repeatedly* occur until all clients are fully-connected, i.e., the not-fully-connected clients set $\mathcal{U} = \emptyset$. At any t , the *payment* of any client j to a site i is defined as t , and the *contribution* is $\max(0, t - c_{ij})$ for the clients in \mathcal{U} and $\max\left(0, \max_q c_{\phi(j^{(q)})j} - c_{ij}\right)$ for the clients in $\mathcal{C} \setminus \mathcal{U}$. As t

increases, the action that a client j connects to a facility of i (x_{ij} is increased by 1) happens under two events: Event 1. j 's payment reaches the connection cost c_{ij} of an already opened facility at i that j is not connected to (implying at this time $y_i > x_{ij}$); Event 2. sum of contributions of all clients to a closed facility at i reaches its opening cost f_i . In particular, if $y_i < R_i$, Event 2 triggers the action that a new facility at i is opened first (y_i is increased by 1). Then any client $j \in \mathcal{C} \setminus \mathcal{U}$ with $\max_q c_{\phi(j^{(q)})j} - c_{ij} > 0$ will switch one of its most expensive connections from $\arg \max_{\phi(j^{(q)})} c_{\phi(j^{(q)})j}$ to i ; and the client in \mathcal{U} with $t - c_{ij} \geq 0$ will connect to i . In addition, for analyzing the approximation ratio of PD, each port $j^{(q)}$ is associated with a dual variable α_j^q which is assigned the time t at which $j^{(q)}$ gets connected. From the algorithm, it should be obvious that $\text{cost}(\mathbf{x}, \mathbf{y}) = \sum_{j \in \mathcal{C}} \sum_{1 \leq q \leq r_j} \alpha_j^q$. Note that Event 2 on i stops occurring once $y_i = R_i$ and this introduces some difficulties to the analysis. To tackle these difficulties, we use an extra variable \hat{x} to store the numbers of the clients' connections when they just become fully-connected.

Algorithm 2 PD: Primal-Dual Algorithm

Input: $\mathcal{F}, \mathcal{C}, \mathbf{f}, \mathbf{c}, \mathbf{r}, \mathbf{R}$. **Output:** (\mathbf{x}, \mathbf{y}) .

Initialization: Set $\mathcal{U} = \mathcal{C}$, $\forall i \in \mathcal{F}, j \in \mathcal{C} : x_{ij}, y_i = 0, p_j = 1$.

while $\mathcal{U} \neq \emptyset$, increase time t uniformly and execute the events below:

- Event 1: $\exists i \in \mathcal{F}, j \in \mathcal{U} : t = c_{ij}$ and $x_{ij} < y_i$.
 Action 1: **set** $\phi(j^{(p_j)}) \leftarrow i, x_{ij} \leftarrow x_{ij} + 1$ and $\alpha_j^{p_j} \leftarrow t$; If $p_j = r_j$, then $\mathcal{U} \leftarrow \mathcal{U} \setminus \{j\}$ and $\hat{x}_{ij} = x_{ij}$, otherwise $p_j \leftarrow p_j + 1$.
- Event 2: $\exists i \in \mathcal{F} : \sum_{j \in \mathcal{U}} \max(0, t - c_{ij}) + \sum_{j \in \mathcal{C} \setminus \mathcal{U}} \max(0, \max_q c_{\phi(j^{(q)})j} - c_{ij}) = f_i$ and $y_i < R_i$.
 Action 2: **set** $y_i \leftarrow y_i + 1$; $\forall j \in \mathcal{C} \setminus \mathcal{U}$ s.t. $\max_q c_{\phi(j^{(q)})j} - c_{ij} > 0$:
set $i_j^* \leftarrow \arg \max_{\phi(j^{(q)})} c_{\phi(j^{(q)})j}, x_{i_j^* j} \leftarrow x_{i_j^* j} - 1, x_{ij} \leftarrow x_{ij} + 1$ and
 $\phi\left(j^{\left(\arg \max_q c_{\phi(j^{(q)})j}\right)}\right) \leftarrow i; \forall j \in \mathcal{U}$ s.t. $t \geq c_{ij}$: **do** Action 1.

Remark 1 If more than one event happen at time t , the algorithm processes all of them in an arbitrary order. Also, the events themselves may repeatedly happen at any t because more than one facilities at a site are allowed to open.

Lemma 6. Algorithm PD computes a feasible solution to the uniform FTRA and runs in $O(n^3 \max_{j \in \mathcal{C}} r_j)$.

Proof. The solution is feasible because (\mathbf{x}, \mathbf{y}) produced from PD is feasible to LP (1). Each iteration of PD at least connects a port of a client, so there are maximum $\sum_{j \in \mathcal{C}} r_j$ iterations. In addition, similar to Theorem 22.4 of [16] and

Theorem 8 of [14] for *UFL*, the client switching in Action 2 dominates the time complexity. In each iteration, the switching takes time $O(n_c n_f)$ to update clients' contributions to other facilities for computing the anticipated times of the events. Hence, the total time is $O\left(\sum_{j \in \mathcal{C}} r_j n_c n_f\right)$, i.e. $O(n^3 \max_{j \in \mathcal{C}} r_j)$.

For a facility location problem modeled by an LP, we observe that the analysis approaches like dual fitting [11] and inverse dual fitting [26,19] are both based on the constraints of the LP. In the following, we develop a simple and systematic constraint-based analysis together with the factor-revealing technique of [11] to derive the ratio of 1.61 for *FTRA*.

The PD algorithm produces a feasible primal solution (\mathbf{x}, \mathbf{y}) to *FTRA* but an infeasible dual solution $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z})$ if we simply let $\alpha_j = \alpha_j^{r_j}$, $\beta_{ij} = \max(0, \alpha_j - c_{ij})$ and $z_i = 0$ in LP (3). This is because although the LP's second constraint holds, the first constraint fails to hold from the events of the algorithm. However, through the steps of properly constructing the dual variables, considering relaxing the feasibilities of the constraints, and proving with the factor-revealing technique that the algorithm can actually ensure these relaxed constraints, the algorithm's approximation factor can be eventually derived. First, we bound the primal solution cost $\text{cost}(\mathbf{x}, \mathbf{y})$ with the dual solution cost $\text{cost}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z})$ using the following dual construction of $\boldsymbol{\alpha}, \mathbf{z}$:

$$\forall i \in \mathcal{F}, j \in \mathcal{C} : \alpha_j = \alpha_j^{r_j}, z_i = \sum_{j \in \mathcal{C}} \pi_{ij} \text{ where } \pi_{ij} = \begin{cases} \frac{x_{ij}(\alpha_j - \alpha_j^{l_{ij}})}{R_i} & \text{if } x_{ij} = R_i \\ 0 & \text{if } x_{ij} < R_i \end{cases}.$$

In the above setting, $\hat{\mathbf{x}}$ stores the primary connection information of the clients after they become fully-connected but before they switch any of their connections; $\forall i \in \mathcal{F}, j \in \mathcal{C} : l_{ij}$ denotes the last port of j connecting to i before switching; $\alpha_j^{l_{ij}}$ is the dual value of the port l_{ij} and $\alpha_j^{r_j}$ is the dual of j 's last port. The other dual variable $\boldsymbol{\beta}$ is to be constructed later in the analysis.

Lemma 7. $\text{cost}(\mathbf{x}, \mathbf{y}) \leq \text{cost}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z})$ where (\mathbf{x}, \mathbf{y}) is the feasible primal solution produced from the PD algorithm and $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z})$ is constructed from above.

Proof.

$$\begin{aligned} \text{cost}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z}) &= \sum_{j \in \mathcal{C}} r_j \alpha_j - \sum_{i \in \mathcal{F}} R_i z_i \\ &= \sum_{j \in \mathcal{C}} \sum_{i \in \mathcal{F}} x_{ij} \alpha_j^{r_j} - \underbrace{\sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} \hat{x}_{ij} (\alpha_j^{r_j} - \alpha_j^{l_{ij}})}_{\text{if } x_{ij} = R_i} - \underbrace{0}_{\text{if } x_{ij} < R_i} \\ &= \sum_{j \in \mathcal{C}} \sum_{i \in \mathcal{F}} \hat{x}_{ij} \alpha_j^{r_j} - \underbrace{\sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} \hat{x}_{ij} (\alpha_j^{r_j} - \alpha_j^{l_{ij}})}_{\text{if } x_{ij} = R_i} \\ &= \underbrace{\sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} \hat{x}_{ij} \alpha_j^{r_j}}_{\text{if } x_{ij} < R_i} + \underbrace{\sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} \hat{x}_{ij} \alpha_j^{l_{ij}}}_{\text{if } x_{ij} = R_i} \end{aligned}$$

$$\text{Hence, } \text{cost}(\mathbf{x}, \mathbf{y}) = \sum_{j \in \mathcal{C}} \sum_{1 \leq q \leq r_j} \alpha_j^q \leq \underbrace{\sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} x_{ij} \alpha_j^{r_j}}_{\text{if } x_{ij} < R_i} + \underbrace{\sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} x_{ij} \alpha_j^{l_{ij}}}_{\text{if } x_{ij} = R_i} = \text{cost}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z}).$$

Second, we exploit the dual constraints of the LP (3). Before going into this, we have the basic definition below.

Definition 1. An algorithm is *bi-factor* (ρ_f, ρ_c) or *single factor* $\max(\rho_f, \rho_c)$ -approximation for FTRA, iff for every instance \mathcal{I} of FTRA and any feasible solution SOL (possibly fractional) of \mathcal{I} with facility cost F_{SOL} and connection cost C_{SOL} , the total cost produced from the algorithm is at most $\rho_f F_{SOL} + \rho_c C_{SOL}$ (ρ_f, ρ_c are both positive constants greater than or equal to one).

In the definition, let any feasible solution be $SOL = (\mathbf{x}'', \mathbf{y}'')$, then $F_{SOL} = \sum_{i \in \mathcal{F}} f_i y_i'', C_{SOL} = \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} c_{ij} x_{ij}''$ and $\text{cost}(\mathbf{x}'', \mathbf{y}'') = F_{SOL} + C_{SOL}$. In the following, we consider the feasibility relaxed dual constraints with the relaxation factors ρ_f and ρ_c :

$$(C6) \quad \forall i \in \mathcal{F}, j \in \mathcal{C} : \alpha_j - \beta_{ij} \leq \rho_c c_{ij}.$$

$$(C7) \quad \forall i \in \mathcal{F} : \sum_{j \in \mathcal{C}} \beta_{ij} \leq \rho_f f_i + z_i.$$

Next, we show that if the dual variables $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z})$ satisfy these relaxed constraints, the corresponding dual cost will be bounded by any feasible primal cost scaled by the factors ρ_f and ρ_c .

Lemma 8. If $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z})$ satisfy (C6) and (C7) while $SOL = (\mathbf{x}'', \mathbf{y}'')$ is any feasible primal solution, then $\text{cost}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z}) \leq \rho_f F_{SOL} + \rho_c C_{SOL}$.

Proof. Since $(\mathbf{x}'', \mathbf{y}'')$ is any feasible solution, all constraints of the LP (2) should hold first. Together with (C6) and (C7), we have:

$$\begin{aligned} \text{cost}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z}) &= \sum_{j \in \mathcal{C}} r_j \alpha_j - \sum_{i \in \mathcal{F}} R_i z_i \\ &\leq \sum_{j \in \mathcal{C}} \sum_{i \in \mathcal{F}} x_{ij}'' \alpha_j - \sum_{i \in \mathcal{F}} y_i'' z_i \\ &\leq \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} [\beta_{ij} y_i'' + (\alpha_j - \beta_{ij}) x_{ij}''] - \sum_{i \in \mathcal{F}} y_i'' z_i \\ &\leq \sum_{i \in \mathcal{F}} (\rho_f f_i + z_i) y_i'' + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} \rho_c c_{ij} x_{ij}'' - \sum_{i \in \mathcal{F}} y_i'' z_i \\ &= \sum_{i \in \mathcal{F}} \rho_f f_i y_i'' + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} \rho_c c_{ij} x_{ij}'' = \rho_f F_{SOL} + \rho_c C_{SOL} \end{aligned}$$

The previous two lemmas and the definition immediately imply the next lemma.

Lemma 9. The PD Algorithm is (ρ_f, ρ_c) -approximation if $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{z})$ satisfy (C6) and (C7).

In order to satisfy (C6), we set $\forall i \in \mathcal{F}, j \in \mathcal{C} : \beta_{ij} = \max(0, \alpha_j - \rho_c c_{ij})$ (because $\beta_{ij} \geq 0$), thereby finishing constructing (α, β, z) . The last step is to find the factors ρ_f and ρ_c s.t. (C7) is also satisfied. The next lemma and corollary are the more specific forms of the previous lemma, after substituting the setting of (α, β, z) into (C7).

Lemma 10. *The PD Algorithm is (ρ_f, ρ_c) -approximation if $\forall i \in \mathcal{F} : \sum_{j \in \mathcal{A}_i} (\alpha_j^{r_j} - \rho_c c_{ij} - \pi_{ij}) \leq \rho_f f_i$ where $\mathcal{A}_i = \{j \in \mathcal{C} \mid \alpha_j^{r_j} \geq \rho_c c_{ij}\}$.*

Proof. After the substitution, (C7) becomes:

$$\begin{aligned} \forall i \in \mathcal{F} : \sum_{j \in \mathcal{C}} (\beta_{ij} - \pi_{ij}) &\leq \rho_f f_i \Rightarrow \\ \forall i \in \mathcal{F} : \sum_{j \in \mathcal{C}} (\max(0, \alpha_j^{r_j} - \rho_c c_{ij}) - \pi_{ij}) &\leq \rho_f f_i \Rightarrow \\ \forall i \in \mathcal{F} : \underbrace{\sum_{j \in \mathcal{C}} (\alpha_j^{r_j} - \pi_{ij} - \rho_c c_{ij})}_{\text{if } \alpha_j^{r_j} \geq \rho_c c_{ij}} - \underbrace{\sum_{j \in \mathcal{C}} \pi_{ij}}_{\text{if } \alpha_j^{r_j} < \rho_c c_{ij}} &\leq \rho_f f_i. \end{aligned}$$

Therefore, since $\pi_{ij} \geq 0$, it is sufficient to prove $\forall i \in \mathcal{F} : \sum_{j \in \mathcal{A}_i} (\alpha_j^{r_j} - \pi_{ij} - \rho_c c_{ij}) \leq \rho_f f_i$ where $\mathcal{A}_i = \{j \in \mathcal{C} \mid \alpha_j^{r_j} \geq \rho_c c_{ij}\}$ to satisfy the original (C7).

If we set $\forall i \in \mathcal{F}, j \in \mathcal{C} : u_{ij} = \alpha_j^{r_j} - \pi_{ij}$, then $u_{ij} = \begin{cases} \alpha_j^{l_{ij}} & \text{if } \hat{x}_{ij} = R_i \\ \alpha_j^{r_j} & \text{if } \hat{x}_{ij} < R_i \end{cases}$ and we have the corollary below.

Corollary 2. *W.l.o.g., for every site i , order the corresponding $n_i = |\mathcal{A}_i|$ clients in $\mathcal{A}_i = \{j \in \mathcal{C} \mid \alpha_j^{r_j} \geq \rho_c c_{ij}\}$ s.t. $u_{i1} \leq \dots \leq u_{in_i}$. Then the PD Algorithm is (ρ_f, ρ_c) -approximation if $\forall i \in \mathcal{F} : \sum_{j=1}^{n_i} (u_{ij} - \rho_c c_{ij}) \leq \rho_f f_i$.*

In addition, for each i , any subset of the clients are ordered from 1 to k_i s.t. $u_{i1} \leq \dots \leq u_{ik_i}$. Now, we proceed the proof to find ρ_f and ρ_c with the following lemmas. These lemmas are needed for the factor-revealing technique and they capture the properties of the PD algorithm for the uniform *FTRA*.

Lemma 11. *For every site i , at time $t = u_{ij} - \epsilon$, $\forall 1 \leq h < j < k_i$ let $\omega_{h,j}^i = \begin{cases} u_{ih} & \text{if } \hat{x}_{ih} = R_i \\ \max_q c_{\phi(h^{(q)})h} & \text{if } \hat{x}_{ih} < R_i \end{cases}$, then $\omega_{h,j}^i \geq \omega_{h,j+1}^i$.*

Proof. If $\hat{x}_{ih} = R_i$, then $\omega_{h,j}^i = \omega_{h,j+1}^i$ (at time $t = u_{i(j+1)} - \epsilon = u_{ih}$). Otherwise, $\hat{x}_{ih} < R_i$ implies $u_{ih} = \alpha_h^{r_h}$, so h is fully-connected at time t since $u_{ih} \leq u_{ij}$. Therefore, $\omega_{h,j}^i \geq \omega_{h,j+1}^i$ because a fully-connected client's ports always reconnect to the sites with less connection cost, so its maximum connection cost will never increase. The lemma follows.

Lemma 12. *For any site i and ordered k_i clients, $\forall 1 \leq j \leq k_i : \sum_{h=1}^{j-1} \max(0, \omega_{h,j}^i - c_{ih}) + \sum_{h=j}^{k_i} \max(0, u_{ij} - c_{ih}) \leq f_i$.*

Proof. For any site i and at time $t = u_{ij} - \epsilon$, if $h < j$ client h 's contribution is set to be $\max(0, \omega_{h,j}^i - c_{ih})$. In particular, from the previous lemma and the setting of u_{ij} , if $x_{ih}^i < R_i$, it implies h is fully-connected at time t and the contribution is $\max(0, \max_q c_{\phi(h(q))h} - c_{ih})$. In addition, if $x_{ih}^i = R_i$ the contribution is $\max(0, \alpha_h^{l_{ih}} - c_{ih})$. Note that under this case, h still might be fully-connected at time t , but because $x_{ih}^i = R_i$ and following the algorithm, its contribution should not be set to $\max(0, \max_q c_{\phi(h(q))h} - c_{ih})$ for ensuring the lemma. On the other hand, if $h \geq j$, h is not fully-connected since $t < \alpha_h^{r_h}$, so we set the contribution to $\max(0, t - c_{ih})$, i.e. $\max(0, u_{ij} - c_{ih})$. From the execution of the algorithm, at any time, the sum of these contributions will not exceed the facility's opening cost at site i , hence the lemma follows.

Lemma 13. *For any site i and clients h, j s.t. $1 \leq h < j \leq k_i : r_h = r_j = r$, then $u_{ij} \leq \omega_{h,j}^i + c_{ij} + c_{ih}$.*

Proof. At time $t = u_{ij} - \epsilon$, if all facilities at site i are already open, then $u_{ij} \leq c_{ij}$ and the lemmas holds. Otherwise, if not all facilities are open, then at time t every client $h < j$ is fully-connected. This is because $u_{ih} \leq u_{ij}$ implies $u_{ih} = \alpha_h^{l_{ih}}$ or $\alpha_h^{r_h}$ at the time t . Since h can only connect to less than R_i facilities at i , this contradicts the condition $x_{ih}^i = R_i$ for the setting of u_{ih} , so $u_{ih} = \alpha_h^{r_h}$. In addition, j itself is not fully-connected at t , whereas h is fully-connected and has already connected to r facilities. There is at least a facility to which h is connected but not j . (This is where we must enforce all clients have the uniform connection r .) Denote this facility (site) by i' , we have $u_{ij} \leq c_{i'j}$ and $\omega_{h,j}^i \geq c_{i'h}$. Lastly, by the triangle inequality of the metric property, $c_{i'j} \leq c_{i'h} + c_{ij} + c_{ih}$ and then we have the lemma.

$$\begin{aligned}
z_k = \text{maximize} \quad & \frac{\sum_{j=1}^k \alpha_j}{f + \sum_{j=1}^k d_j} \\
\text{subject to} \quad & \forall 1 \leq j < k : \alpha_j \leq \alpha_{j+1} \\
& \forall 1 \leq h < j < k : r_{h,j} \geq r_{h,j+1} \\
& \forall 1 \leq h < j \leq k : \alpha_j \leq r_{h,j} + d_h + d_j \\
& 1 \leq j \leq k : \sum_{h=1}^{j-1} \max(r_{h,j} - d_h, 0) + \sum_{h=j}^k \max(\alpha_j - d_h, 0) \leq f \\
& 1 \leq h \leq j < k : \alpha_j, d_j, f, r_{h,j} \geq 0
\end{aligned}$$

Consider the above factor-revealing program series (25) of [11]. If we let $k = k_i, \alpha_j = u_{ij}, r_{h,j} = \omega_{h,j}^i, f = f_i, d_j = c_{ij}$, from the previous property lemmas it is clear that $u_{ij}, \omega_{h,j}^i, f_i$ and c_{ij} constitute a feasible solution. Also, from Lemma 5.4 and Theorem 8.3 of [11], and Lemma 4 of [22] we can directly get $\forall i \in \mathcal{F} : \sum_{j=1}^{k_i} (u_{ij} - 1.61c_{ij}) \leq 1.61f_i, \sum_{j=1}^{k_i} (u_{ij} - 1.78c_{ij}) \leq 1.11f_i$ and $\sum_{j=1}^{k_i} (u_{ij} - 2c_{ij}) \leq f_i$. Furthermore, because $n_i = |\mathcal{A}_i|$ and k_i represents the

number of any subset of the clients, Lemma 6 and Corollary 2 directly lead to the following theorem.

Theorem 3. *Algorithm PD is 1.61-, (1.11, 1.78)- and (1,2)-approximation in time $O(n^3 \max_{j \in \mathcal{C}} r_j)$ for the uniform FTRA.*

Algorithm 3 SPD: Speed-up of Primal-Dual Algorithm

Input: $\mathcal{F}, \mathcal{C}, \mathbf{f}, \mathbf{c}, \mathbf{r}, \mathbf{R}$. **Output:** (\mathbf{x}, \mathbf{y}) .

Initialization: Set $\mathcal{U} = \mathcal{C}$, $\forall i \in \mathcal{F}, j \in \mathcal{C} : x_{ij}, y_i = 0, FC_j = 0$.

while $\mathcal{U} \neq \emptyset$, increase time t uniformly and execute the events below:

- Event 1: $\exists i \in \mathcal{F}, j \in \mathcal{U}$ s.t. $t = c_{ij}$ and $x_{ij} < y_i$.
 Action 1-a: $ToC \leftarrow \min(y_i - x_{ij}, r_j - FC_j)$;
 Action 1-b: **set** $x_{ij} \leftarrow x_{ij} + ToC$ and $FC_j \leftarrow FC_j + ToC$;
 Action 1-c: If $FC_j = r_j$ then $\mathcal{U} \leftarrow \mathcal{U} \setminus \{j\}$.
- Event 2: $\exists i \in \mathcal{F} : \sum_{j \in \mathcal{U}} \max(0, t - c_{ij}) + \sum_{j \in \mathcal{C} \setminus \mathcal{U}} \max(0, \max_{i' \in \mathcal{F} \& x_{i'j} > 0} c_{i'j} - c_{ij}) = f_i$ and $y_i < R_i$.
 Action 2-a: $\mathcal{U}_i \leftarrow \{j \in \mathcal{U} \mid t - c_{ij} \geq 0\}$ and $NC \leftarrow \min_{j \in \mathcal{U}_i} (r_j - FC_j)$;
 Action 2-b: $\mathcal{S}_i \leftarrow \{j \in \mathcal{C} \setminus \mathcal{U} \mid \max_{i' \in \mathcal{F} \& x_{i'j} > 0} c_{i'j} - c_{ij} > 0\}$, $\forall j \in \mathcal{S}_i : i_j^* \leftarrow \arg \max_{i' \in \mathcal{F} \& x_{i'j} > 0} c_{i'j}$ and $NS \leftarrow \min_{j \in \mathcal{S}_i} x_{i_j^* j}$;
 Action 2-c: **set** $ToC \leftarrow \min(NC, NS, R_i - y_i)$ and $y_i \leftarrow y_i + ToC$; $\forall j \in \mathcal{S}_i : x_{i_j^* j} \leftarrow x_{i_j^* j} - ToC$ and $x_{ij} \leftarrow x_{ij} + ToC$; $\forall j \in \mathcal{U}_i : \mathbf{do}$ Action 1-b;
 Action 2-d: $\forall j \in \mathcal{U}_i : \mathbf{do}$ Action 1-c.

Remark 2 *For the convenience of analysis, sequential actions of the events are separated as above. If more than one event happens at the same time, the algorithm process Event 2 first so that no repeated events are needed.*

The previous PD algorithm runs in pseudo-polynomial time depending on r_j . With the speed-up heuristic described in the following, the algorithm can then change to an essentially identical algorithm SPD (Algorithm 3) which is strongly polynomial. In the heuristic, (\mathbf{x}, \mathbf{y}) is able to increase at a faster rate rather than 1, through combining the repeated events into a single event and thereby reducing the total number of events to process. In particular, for Event 2, once a facility of a site i is opened and connected with a group of clients' ports, according to the PD algorithm, additional facilities at i may subsequently open and connect with this group of clients' other ports until their sum of contributions (SOC) becomes insufficient to pay f_i , or $y_i = R_i$. The SOC is not enough any more if a client in \mathcal{U} appears to be fully-connected, so $\sum_{j \in \mathcal{U}} \max(0, t - c_{ij})$ will decrease, or the most expensive connection of a client in $\mathcal{C} \setminus \mathcal{U}$ differs (after switching), in

this case $\sum_{j \in \mathcal{C} \setminus \mathcal{U}} \max \left(0, \max_{i' \in \mathcal{F} \& x_{i'j} > 0} c_{i'j} - c_{ij} \right)^2$ will decrease. Similarly, for Event 1, once a client j 's port starts to connect to an already opened facility at a site i , its other ports may get connected to i at the same time until either there are no remaining open facilities at i or j reaches r_j connections.

Formally in the SPD Algorithm, FC_j denotes the number of established connections of client j and ToC the total number of connections decided to make according to the heuristic. The incremental rate of (\mathbf{x}, \mathbf{y}) can then be determined by ToC instead of 1. Moreover, in the more complicated Event 2 on a site i , NC denotes the maximum number of connections to make until one of the clients in \mathcal{U} gets fully-connected, and NS the maximum number of switches until the most expensive connection of a client in $\mathcal{C} \setminus \mathcal{U}$ changes. Therefore, ToC is calculated as $\min(NC, NS, R_i - y_i)$, the maximum number of connections until the SOC becomes insufficient or $y_i = R_i$.

Lemma 14. *Algorithm SPD computes a feasible solution to the FTTRA and runs in $O(n^4)$.*

Proof. The solution is feasible because SPD is essentially the same as PD except the implementation of the speed-up heuristic. Also with this heuristic, the number of Event 1 is at most $n_f n_c$ because for any client j and site i only when $t = c_{ij}$, j exhaustively gets connected to open facilities at site i , and there are n_f sites and n_c clients in total. So the numbers of both Action 1-a and 1-b are bounded by $n_f n_c$, while Action 1-c is bounded by n_c since there are n_c clients to be connected.

Moreover, the number of Event 2 can be bounded by $n_c n_f$ instead of $\sum_{i \in \mathcal{F}} R_i$. This is because each Event 2 will cause at least one of the following 3 cases: (1) a client j in \mathcal{U} becomes fully-connected; (2) a client j in $\mathcal{C} \setminus \mathcal{U}$ switches all of its most expensive connections; (3) a site open all its facilities. It is easy to see that there are maximum n_c and n_f cases (1) and (3) respectively, so we are left to bound the number of case (2). For this case, it is important to observe that any client j has at most n_f possible sets of connections where connections in each set associate to the same site. So there are at most $n_c n_f$ such possible sets in total, and each case (2) at least removes a possible set, i.e. at least a client's connections have one less possible site to further switch to, since clients only switch to cheaper connections. Therefore, there are at most $n_c n_f$ case (2) and $(n_c + n_f + n_c n_f)$ Event 2. So the numbers of Action 2-a, 2-b and 2-c are bounded by $O(n_c n_f)$ while Action 2-d is included in Action 1-c. Lastly, as in the PD algorithm, the switching action Action 2-c dominates the time complexity of all actions which takes $O(n_c n_f)$, hence the total time is $O(n_c^2 n_f^2)$.

The algorithm computes the same solution as PD, so we obtain the following theorem.

² For simplicity of the algorithm description, we replace the term $\sum_{j \in \mathcal{C} \setminus \mathcal{U}} \max \left(0, \max_q c_{\phi(j(q))j} - c_{ij} \right)$ in the PD algorithm with essentially the same term here.

Theorem 4. *Algorithm SPD is 1.61-, (1.11, 1.78)- and (1,2)-approximation in time $O(n^4)$ for the uniform FTRA.*

Algorithm 4 SGA: Speed-up of Greedy Augmentation

Input: $\mathcal{F}, \mathcal{C}, \mathbf{f}, \mathbf{c}, \mathbf{r}, \mathbf{R}, (\mathbf{x}, \mathbf{y})$. **Output:** (\mathbf{x}, \mathbf{y}) .

Initialization:

for $j \in \mathcal{C}$ //optimize the total connection cost first

for $i \in \mathcal{F}$ and $y_i > 0$, in the increasing order of distances w.r.t j

$x_{ij} \leftarrow \min(r_j, y_i)$

$r_j \leftarrow r_j - x_{ij}$

set residual vector $\bar{\mathbf{y}} \leftarrow \mathbf{R} - \mathbf{y}$ //for detecting the case y_i reaches R_i

set $CC \leftarrow \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} c_{ij} x_{ij}$ as the current total connection cost

invoke calculateGain

while $\max_{i \in \mathcal{F}} \text{gain}(i) > 0$: //if $\text{gain}(i) > 0$, then $\bar{y}_i > 0$ from the calculateGain function

pick $i^* = \arg \max_{i \in \mathcal{F}} \frac{\text{gain}(i)}{f_i}$

$\mathcal{S}_i \leftarrow \left\{ j \in \mathcal{C} \mid \max_{i' \in \mathcal{F} \& \& x_{i'j} > 0} c_{i'j} - c_{i^*j} > 0 \right\}$

$\forall j \in \mathcal{S}_i : i_j^* \leftarrow \arg \max_{i' \in \mathcal{F} \& \& x_{i'j} > 0} c_{i'j}$

$NS \leftarrow \min_{j \in \mathcal{S}_i} x_{ij^*}, ToC \leftarrow \min(NS, \bar{y}_i)$

set $y_{i^*} \leftarrow y_{i^*} + ToC$

$\Delta \leftarrow 0$ // Δ stores the total decrease in the connection cost after all switches

for $j \in \mathcal{S}_i$

$\Delta \leftarrow \Delta + ToC \cdot \left(\max_{i' \in \mathcal{F} \& \& x_{i'j} > 0} c_{i'j} - c_{i^*j} \right)$

set $x_{ij^*} \leftarrow x_{ij^*} - ToC$ and $x_{i^*j} \leftarrow x_{i^*j} + ToC$

set $CC \leftarrow CC - \Delta$

update $\bar{\mathbf{y}}$

invoke calculateGain

function calculateGain

for $i \in \mathcal{F}$

$C_i \leftarrow CC$ //for each i , C_i stores the total connection cost after connections are switched to i

$\text{gain}(i) \leftarrow 0$

if $\bar{y}_i > 0$

for $j \in \mathcal{C}$

if $\max_{i' \in \mathcal{F} \& \& x_{i'j} > 0} c_{i'j} > c_{ij}$

$C_i \leftarrow C_i - \max_{i' \in \mathcal{F} \& \& x_{i'j} > 0} c_{i'j} + c_{ij}$

$\text{gain}(i) \leftarrow CC - C_i - f_i$

In order to further achieve the factor of 1.52 in strongly polynomial time that matches the best result [25] for the uniform *FTFL*, it is necessary to apply the cost scaling and greedy augmentation (GA) techniques [25,8] for *FTFL* to *FTRA*. However, like in [18,27] the difficulty encountered is the application of greedy augmentation (GA) in polynomial time, since the naive way of treat-

ing an $FTRA/FTRA_\infty$ instance as an equivalent $FTFL$ instance and then directly applying GA after cost scaling will result in weakly polynomial or pseudo-polynomial time algorithms, depending on whether using the instance shrinking technique in the previous section or not.

Nevertheless, if GA is applied with another similar speed-up heuristic, it changes to the algorithm SGA (Algorithm 4) which runs in strongly polynomial time. Before describing SGA, we take a brief look at GA in [8] for $FTFL$. It defines $gain(i)$ of a facility i to be the decrease in total cost (decrease in total connection cost minus increase in facility cost of i) of the solution after adding a facility i to open and connecting clients to their closest facilities. Note that once a set of open facilities are fixed, the total connection cost can be easily computed since every client simply chooses these facilities in increasing order of distance. GA then iteratively picks the facility with the largest gain ratio $\frac{gain(i)}{f_i}$ to open until there is no facility i with $gain(i) > 0$ left. On the other hand, SGA computes $gain(i)$ in the same way as GA. The difference is in $FTRA$ there are $\sum_{i \in \mathcal{F}} R_i$ facilities in total, therefore it is slow to consider one facility at a time (in each iteration of SGA). Fortunately, there is also a speed-up heuristic: because all facilities at a site i has $gain(i)$, once a facility at site i_m with $\max_i \frac{gain(i)}{f_i}$ is selected to open, additional facilities at i_m may also open at the same time until either (1) this maximum $gain(i_m)$ reduces due to insufficient decrease in the total connection cost; or (2) y_i reaches R_i . Moreover, (1) happens once a client has appeared to switch all of its most expensive connections to i_m , which is similar to the switching case in the previous algorithm SPD.

Formally in the SGA algorithm, CC denotes the current total connection cost and C_i the connection cost after i is opened and client connections are switched. The calculateGain function computes $gain(i)$ and the while loop implements GA with the described heuristic. In each loop iteration, for updating CC , Δ stores the total decrease in the connection cost after client switching. Following the heuristic, ToC and NS are defined similarly as in the SPD algorithm. Note that in the initialization phase of SGA, the total connection cost is optimized first so that every client connects to its closest facilities. This is to ensure that in every iteration only the client connections with the largest costs need to be considered in computing the best possible connection cost C_i .

Lemma 15. *Algorithm SGA runs in $O(n^4)$ for $FTRA$.*

Proof. Each iteration of the while loop runs in $O(n_c n_f)$ due to the calculateGain function. Now, we bound the the total number of iterations. Similar to the runtime analysis of the algorithm SPD (c.f. Lemma 14), in SGA once a site i_m with the maximum gain is chosen, SGA opens the facilities at i_m until either R_{i_m} is reached, or a client has appeared to switch all of its most expensive connections, causing reduced maximum gain. Further, there are at most n_f chances to reach R_{i_m} and $n_c n_f$ possible sets of connections for all clients. Since clients also only switch to cheaper connections, there are maximum $(n_f + n_c n_f)$ iterations. The total time is therefore $O(n_c^2 n_f^2)$.

Now the important observation/trick for the analysis is that applying SGA to an $FTRA/FTRA_\infty$ instance (with solution) obtains essentially the *same solution* (also the same cost) as treating this instance as an equivalent $FTFL$ instance (by naively splitting sites) and then directly applying GA. The difference is, with the speed-up heuristic, SGA is able to arrive at this solution faster, in strongly polynomial time. The observation then implies that SGA alone improves the 3.16-approximation result of [27] for the general $FTRA_\infty$ to 2.408 in polynomial time using the GA results [8] for $FTFL$. Similarly, for the uniform $FTRA$, SGA combined with cost scaling further improves on the (1.11, 1.78)-approximation algorithm SPD according to the results of [25] for the uniform $FTFL$.

Theorem 5. *The uniform $FTRA$ can be approximated with a factor of 1.52 in time $O(n^4)$.*

6 The Uniform $KFTRA$

Lastly, we consider the Constrained Fault-Tolerant k -Resource Allocation ($KFTRA$) problem and show its uniform case achieves an approximation ratio of 4. In this important variant of $FTRA$, there is an additional constraint that at most k facilities ($\max_{j \in \mathcal{C}} r_j \leq k$ and $k \leq \sum_{i \in \mathcal{F}} R_i$) across the sites can be opened as resources. This problem has the following formulation.

$$\begin{aligned}
& \text{minimize} && \sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} c_{ij} x_{ij} \\
& \text{subject to} && \forall j \in \mathcal{C} : \sum_{i \in \mathcal{F}} x_{ij} \geq r_j \\
& && \forall i \in \mathcal{F}, j \in \mathcal{C} : y_i - x_{ij} \geq 0 \\
& && \sum_{i \in \mathcal{F}} y_i \leq k \\
& && \forall i \in \mathcal{F} : y_i \leq R_i \\
& && \forall i \in \mathcal{F}, j \in \mathcal{C} : x_{ij}, y_i \in \mathbb{Z}^+
\end{aligned} \tag{4}$$

Its LP-relaxation (primal LP) and dual LP are:

$$\begin{aligned}
& \text{minimize} && \sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} c_{ij} x_{ij} \\
& \text{subject to} && \forall j \in \mathcal{C} : \sum_{i \in \mathcal{F}} x_{ij} \geq r_j \\
& && \forall i \in \mathcal{F}, j \in \mathcal{C} : y_i - x_{ij} \geq 0 \\
& && \sum_{i \in \mathcal{F}} y_i \leq k \\
& && \forall i \in \mathcal{F} : y_i \leq R_i \\
& && \forall i \in \mathcal{F}, j \in \mathcal{C} : x_{ij}, y_i \geq 0
\end{aligned} \tag{5}$$

$$\begin{aligned}
& \text{maximize} && \sum_{j \in \mathcal{C}} r_j \alpha_j - \sum_{i \in \mathcal{F}} R_i z_i - k\theta \\
& \text{subject to} && \forall i \in \mathcal{F} : \sum_{j \in \mathcal{C}} \beta_{ij} \leq f_i + z_i + \theta \\
& && \forall i \in \mathcal{F}, j \in \mathcal{C} : \alpha_j - \beta_{ij} \leq c_{ij} \\
& && \forall i \in \mathcal{F}, j \in \mathcal{C} : \alpha_j, \beta_{ij}, z_i, \theta \geq 0
\end{aligned} \tag{6}$$

It is clear that $KFTRA$ generalizes the well studied $KUFL$ [14,11] and $KFTFL$ [25] problems. In the following, besides adapting the algorithms and analyses therein, we also develop a greedy pairing (GP) procedure which in

polynomial time constructs paired and unpaired sets of facilities from sites for randomly opening them afterwards.

Algorithm Description. The algorithm PK (Algorithm 5) consists of three sequential procedures: Binary Search (BS), Greedy Pairing (GP) and Randomized Rounding (RR). BS utilizes the previous (1, 2)-approximation algorithm SPD (Algorithm 3) for *FTRA* with the *modified* input facility cost $2(f_i + \theta)$, i.e. the cost is increased by θ first and then scaled by 2. As we will see later in the analysis, this modification is necessary for two reasons: 1) the Lagrangian relaxation of *KFTRA* is *FTRA*; 2) the scaling of the facility cost enables us to build on the approximation ratio (1, 2) of *FTRA* for getting the ratio of *KFTRA*. For simplicity, let $\text{SPD}(\theta, \lambda)$ denote the parameterized SPD algorithm with the input facility cost perturbing factor θ and scaling factor λ , so $\text{SPD}(0, 1)$ produces the same solution as SPD. From LP (1) and (4), it is clear that SPD produces an almost feasible integral solution to *KFTRA* except that it has to guarantee at most k facilities to open ($\sum_{i \in \mathcal{F}} y_i \leq k$) from all sites. This guarantee might not be even possible, but fortunately we can use $\text{SPD}(\theta, \lambda)$ to get two solutions $(\mathbf{x}_s, \mathbf{y}_s)$ and $(\mathbf{x}_l, \mathbf{y}_l)$ with the small one having $\sum_{i \in \mathcal{F}} y_{s,i} = k_s < k$ facilities open and the large $\sum_{i \in \mathcal{F}} y_{l,i} = k_l > k$. A convex combination of these two solutions is able to give a feasible *fractional* solution $(\mathbf{x}', \mathbf{y}')$ to *KFTRA* instead, i.e. $(\mathbf{x}', \mathbf{y}') = a(\mathbf{x}_s, \mathbf{y}_s) + b(\mathbf{x}_l, \mathbf{y}_l)$ with $a + b = 1$ and $ak_s + bk_l = k$. The solutions can be obtained by binary searching two values of θ (θ_1 and θ_2) over the interval $[0, \frac{n c_{\max}}{\lambda}]$ where $c_{\max} = \max_{i \in \mathcal{F}, j \in \mathcal{C}} c_{ij}$ and invoking $\text{SPD}(\theta_1, \lambda)$ and $\text{SPD}(\theta_2, \lambda)$. This specific interval is chosen because as the value of θ increases, the number of open facilities from $\text{SPD}(\theta, \lambda)$ will decrease. So if $\theta = \frac{n c_{\max}}{\lambda}$, the algorithm will only open the minimum number of $\max_{j \in \mathcal{C}} r_j$ facilities.³ Moreover, as shown later, if θ_1 and θ_2 become sufficiently close ($\epsilon = \frac{c_{\min}}{8N^2}$ where c_{\min} is the smallest positive connection cost and $N = \sum_{i \in \mathcal{F}} R_i$) in BS, the approximation ratio of SPD is almost preserved for building a ratio for *KFTRA*.

However, for a feasible *integral* solution (\mathbf{x}, \mathbf{y}) with k open facilities, the algorithm instead relies our *efficient* GP procedure. Based on the solution vectors \mathbf{y}_s and \mathbf{y}_l obtained from BS, GP splits the vector \mathbf{y}_l into \mathbf{y}_p and $\tilde{\mathbf{y}}_p$ s.t. $\mathbf{y}_l = \mathbf{y}_p + \tilde{\mathbf{y}}_p$ and $\sum_{i \in \mathcal{F}} y_{s,i} = \sum_{i \in \mathcal{F}} y_{p,i} = k_s$. Note that each of these integral vectors represents the facility opening amounts of all sites. To be precise, GP greedily constructs the paired (\mathbf{y}_p) and unpaired facilities ($\tilde{\mathbf{y}}_p$) from \mathbf{y}_l against the small solution \mathbf{y}_s . In particular, it first pairs the facilities of the corresponding sites in \mathbf{y}_s and \mathbf{y}_l (both sites with open facilities) and records the pairing result in \mathbf{y}_p . Next, for each left unpaired site i in $\tilde{\mathbf{y}}_s$ in arbitrary order, GP exhaustively pairs the facilities at i with the facilities of the unpaired sites in $\tilde{\mathbf{y}}_l$ in the order of closest to i . In this pairing step, \mathbf{y}_p is updated accordingly. At the end, $\tilde{\mathbf{y}}_p$ is simply set to be $\mathbf{y}_l - \mathbf{y}_p$. To be more precise, we consider a simple example with $\mathbf{y}_s = [3, 2, 0, 2]$ and $\mathbf{y}_l = [2, 0, 5, 3]$ from BS before running GP. After the

³ We noticed that the binary search interval $[0, nrc_{\max}]$ (c.f. the third paragraph of Section 7 of [25]) for *KFTFL* can be reduced to $[0, \frac{n c_{\max}}{2}]$, because once the minimum number of $\max_{j \in \mathcal{C}} r_j$ facilities are opened and all facility costs are at least $n c_{\max}$, from the primal-dual algorithm, all clients are already fully-connected.

Algorithm 5 PK: Procedures for *KFTRA*

Input: A *KFTRA* instance $\langle \mathcal{F}, \mathcal{C}, f_i, c_{ij}, r_j, R_i, k \rangle$. **Output:** (\mathbf{x}, \mathbf{y})

Initialization: $\mathbf{x} \leftarrow \mathbf{0}, \mathbf{y} \leftarrow \mathbf{0}$

Procedure 1: Binary Search (BS)

$\theta_1 = 0, \theta_2 = \frac{n_c \max_{i \in \mathcal{F}, j \in \mathcal{C}} c_{ij}}{2}$

while $\theta_2 - \theta_1 > \epsilon$ **do**:

$mid = \frac{\theta_2 + \theta_1}{2}$

invoke SPD with $\langle \mathcal{F}, \mathcal{C}, 2(f_i + mid), c_{ij}, r_j, R_i \rangle$ and output $(\mathbf{x}_{mid}, \mathbf{y}_{mid})$

$k_{mid} = \sum_{i \in \mathcal{F}} y_{mid,i}$

if $k_{mid} < k$

set $\theta_2 = mid$

else if $k_{mid} > k$

set $\theta_1 = mid$

else

return mid //if here reached, all procedures afterwards can be safely ignored

invoke SPD with $\langle \mathcal{F}, \mathcal{C}, 2(f_i + \theta_1), c_{ij}, r_j, R_i \rangle$ and output $(\mathbf{x}_l, \mathbf{y}_l)$

invoke SPD with $\langle \mathcal{F}, \mathcal{C}, 2(f_i + \theta_2), c_{ij}, r_j, R_i \rangle$ and output $(\mathbf{x}_s, \mathbf{y}_s)$

$k_l = \sum_{i \in \mathcal{F}} y_{l,i} > k$ and $k_s = \sum_{i \in \mathcal{F}} y_{s,i} < k$

Procedure 2: Greedy Pairing (GP)

//vectors representing numbers of constructed paired and unpaired facilities in \mathbf{y}_l

set $\mathbf{y}_p \leftarrow \mathbf{0}, \bar{\mathbf{y}}_p \leftarrow \mathbf{0}$

for $i \in \mathcal{F}$

if $y_{s,i} > 0$ and $y_{l,i} > 0$

$y_{p,i} \leftarrow y_{p,i} + \min(y_{s,i}, y_{l,i})$

//updating vectors and store them in $\hat{\mathbf{y}}_s$ and $\hat{\mathbf{y}}_l$ for the next pairing steps

$\hat{y}_{s,i} \leftarrow y_{s,i} - \min(y_{s,i}, y_{l,i})$

$\hat{y}_{l,i} \leftarrow y_{l,i} - \min(y_{s,i}, y_{l,i})$

for $i \in \mathcal{F}$ in arbitrary order

if $\hat{y}_{s,i} > 0$

for $i' \in \mathcal{F} \setminus i$ in the order of closest to i

if $\hat{y}_{l,i'} > 0$

$y_{p,i'} \leftarrow y_{p,i'} + \min(\hat{y}_{s,i}, \hat{y}_{l,i'})$

$\hat{y}_{s,i} \leftarrow \hat{y}_{s,i} - \min(\hat{y}_{s,i}, \hat{y}_{l,i'})$

$\hat{y}_{l,i'} \leftarrow \hat{y}_{l,i'} - \min(\hat{y}_{s,i}, \hat{y}_{l,i'})$

$\bar{\mathbf{y}}_p \leftarrow \mathbf{y}_l - \mathbf{y}_p$ //at this time $\sum_{i \in \mathcal{F}} y_{p,i} = k_s$ and $\sum_{i \in \mathcal{F}} \bar{y}_{p,i} = k_l - k_s$

Procedure 3: Randomized Rounding (RR)

choose probabilities $a = \frac{k_l - k}{k_l - k_s}$ and $b = \frac{k - k_s}{k_l - k_s}$ so $ak_s + bk_l = k$ and $a + b = 1$

set $\mathbf{y} \leftarrow \mathbf{y}_s$ with probability a and $\mathbf{y} \leftarrow \mathbf{y}_p$ with probability $b = 1 - a$ //disjoint cases both open k_s facilities

select a random subset of $k - k_s$ facilities to open from $\bar{\mathbf{y}}_p$ and add these to \mathbf{y} //at this time $\sum_{i \in \mathcal{F}} y_i = k$ and each facility in \mathbf{y}_l is opened with probability b

//connects each client j to its closest r_j opened facilities

for $j \in \mathcal{C}$

for $i \in \mathcal{F}$ in the order of closest to j

$x_{ij} \leftarrow \min(r_j, y_i)$

$r_j \leftarrow r_j - x_{ij}$

first pairing step, $\mathbf{y}_p = [2, 0, 0, 2]$, $\mathbf{y}_s = [1, 2, 0, 0]$ and $\mathbf{y}_l = [0, 0, 5, 1]$. Now for simplicity, we assume that the distance between sites i and j is $|i - j|$ where we follow the ascending order of indices j 's in resolving the ties of the closest distances. Therefore, after the second step, $\mathbf{y}_p = [2, 0, 3, 2]$, $\mathbf{y}_s = [0, 0, 0, 0]$ and $\mathbf{y}_l = \mathbf{y}_p = [0, 0, 2, 1]$, since both the unpaired $y_{s,1}$ and $y_{s,2}$ (1-based index) are paired to the closest unpaired $y_{l,3}$.

Based on the \mathbf{y}_s , \mathbf{y}_p and \mathbf{y}_l obtained, the last procedure RR then randomly opens k facilities in a way that ensures the expected facility opening cost in \mathbf{y} is the *same* as the cost of the opening facilities in the convex combination solution \mathbf{y}' . Finally, according to \mathbf{y} , RR connects each client j to its closest r_j opened facilities via updating \mathbf{x} .

Algorithm Analysis. The basic idea of the analysis is to first bound $\text{cost}(\mathbf{x}', \mathbf{y}')$ by $\text{cost}(\boldsymbol{\alpha}', \boldsymbol{\beta}', \mathbf{z}', \theta')$ where $(\boldsymbol{\alpha}', \boldsymbol{\beta}', \mathbf{z}', \theta')$ is a feasible dual solution to LP (6). Then we bound the expected total cost $\text{cost}(\mathbf{x}, \mathbf{y})$ with $\text{cost}(\mathbf{x}', \mathbf{y}')$ to further establish the approximation ratio ρ s.t. $\text{cost}(\mathbf{x}, \mathbf{y}) \leq \rho \text{cost}(\boldsymbol{\alpha}', \boldsymbol{\beta}', \mathbf{z}', \theta')$. Finally, by the weak duality theorem, $\text{cost}(\mathbf{x}, \mathbf{y}) \leq \rho \text{cost}(\mathbf{x}_k^*, \mathbf{y}_k^*)$ where $(\mathbf{x}_k^*, \mathbf{y}_k^*)$ is the optimal fractional solution to *KFTRA* (displayed as LP (5)).

For the first step, we focus on analyzing the BS procedure to bound $\text{cost}(\mathbf{x}', \mathbf{y}')$ by $\text{cost}(\boldsymbol{\alpha}', \boldsymbol{\beta}', \mathbf{z}', \theta')$. Suppose $\text{SPD}(\theta, 2)$ produces the primal solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ with \tilde{k} open facilities. We let the cost of $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ w.r.t. the *original* input instance be $\text{cost}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = \tilde{F} + \tilde{C}$, where in the separate costs (\tilde{F}, \tilde{C}) , $\tilde{F} = \sum_{i \in \mathcal{F}} f_i \tilde{y}_i$ is the total facility cost and $\tilde{C} = \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} c_{ij} \tilde{x}_{ij}$ is the connection cost. Similarly, w.r.t. the modified instance, the cost is $\text{cost}'(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = 2(\tilde{F} + \tilde{k}\theta) + \tilde{C}$. From the analysis (cf. the paragraph before Theorem 3) of the factor revealing program of the PD algorithm, for $\text{SPD}(\theta, 2)$, we get $\forall i \in \mathcal{F} : \sum_{j \in \mathcal{C}'} (u_{ij} - 2c_{ij}) \leq 2(f_i + \theta)$ where $\mathcal{C}' \subseteq \mathcal{C}$, i.e.,

$$\sum_{j \in \mathcal{C}} \tilde{\alpha}_j - 2 \sum_{j \in \mathcal{C}} c_{ij} - \tilde{z}_i - 2(f_i + \theta) \leq 0 \quad (7)$$

where $(\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\beta}}, \tilde{\mathbf{z}})$ is the corresponding constructed dual values of $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ from the PD algorithm. Further, from Lemma 7, we have a bound for $\text{cost}'(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$, i.e.,

$$2(\tilde{F} + \tilde{k}\theta) + \tilde{C} \leq \sum_{j \in \mathcal{C}} r_j \tilde{\alpha}_j - \sum_{i \in \mathcal{F}} R_i \tilde{z}_i \quad (8)$$

. Note that the dual solution $(\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\beta}}, \tilde{\mathbf{z}})$ is used only in the analysis. Also, because SPD only speeds up PD by combining its events, we can use the dual solution produced from PD for analyzing SPD. If we set $(\tilde{\boldsymbol{\alpha}}', \tilde{\mathbf{z}}', \tilde{\theta}') = (\frac{\tilde{\boldsymbol{\alpha}}}{2}, \frac{\tilde{\mathbf{z}}}{2}, \theta)$ and $\forall i \in \mathcal{F}, j \in \mathcal{C} : \tilde{\beta}'_{ij} = \tilde{\alpha}'_j - c_{ij}$, the inequality (7) then becomes $\sum_{j \in \mathcal{C}} \tilde{\beta}'_{ij} \leq f_i + \tilde{z}'_i + \tilde{\theta}'$, implying $(\tilde{\boldsymbol{\alpha}}', \tilde{\boldsymbol{\beta}}', \tilde{\mathbf{z}}', \tilde{\theta}')$ is a feasible dual solution to LP (6). Furthermore, (8) becomes

$$2\tilde{F} + \tilde{C} \leq 2 \left(\sum_{j \in \mathcal{C}} r_j \tilde{\alpha}'_j - \sum_{i \in \mathcal{F}} R_i \tilde{z}'_i - \tilde{k} \tilde{\theta}' \right) \quad (9)$$

. The analysis here reveals the Lagrangian relation between *KFTRA* and *FTRA* from the dual perspective, whereas the Lagrangian relaxation framework (cf. Section 3.6 of [14]) starts from the primal. Therefore, if $\tilde{k} = k$, $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ is 2-approximation from the inequality (9), the bound $\text{cost}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) < 2\tilde{F} + \tilde{C}$ and the feasibilities of $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ and $(\tilde{\alpha}', \tilde{\beta}', \tilde{z}', \tilde{\theta}')$. However, as mentioned before, we may never encounter the situation $\tilde{k} = k$. Instead, the BS procedure finds θ_1 and θ_2 until $\theta_2 - \theta_1 \leq \epsilon = \frac{c_{\min}}{8N^2}$. It then runs $\text{SPD}(\theta_1, 2)$ to obtain the solution $(\mathbf{x}_l, \mathbf{y}_l)$ with $k_l > k$ and the cost (F_l, C_l) w.r.t. the original instance; and $\text{SPD}(\theta_2, 2)$ to get the solution $(\mathbf{x}_s, \mathbf{y}_s)$ with $k_s < k$ and (F_s, C_s) . Hence, from (9) we have

$$2F_l + C_l \leq 2 \left(\sum_{j \in \mathcal{C}} r_j \alpha'_{l,j} - \sum_{i \in \mathcal{F}} R_i z'_{l,i} - k_l \theta_1 \right) \quad (10)$$

and

$$2F_s + C_s \leq 2 \left(\sum_{j \in \mathcal{C}} r_j \alpha'_{s,j} - \sum_{i \in \mathcal{F}} R_i z'_{s,i} - k_s \theta_2 \right) \quad (11)$$

where $(\alpha'_l, \beta'_l, z'_l)$ and $(\alpha'_s, \beta'_s, z'_s)$ are constructed as $(\tilde{\alpha}', \tilde{\beta}', \tilde{z}')$ to be feasible duals.

Now we are ready to bound $\text{cost}(\mathbf{x}', \mathbf{y}')$ by $\text{cost}(\alpha', \beta', z', \theta')$. The proof of the following lemma builds on the idea of Lemma 9 in [14] for k -median.

Lemma 16. $\text{cost}(\mathbf{x}', \mathbf{y}') < (2 + \frac{1}{N}) F' + C' \leq (2 + \frac{1}{N}) \text{cost}(\alpha', \beta', z', \theta')$, where $N = \sum_{i \in \mathcal{F}} R_i$, $(\mathbf{x}', \mathbf{y}') = a(\mathbf{x}_s, \mathbf{y}_s) + b(\mathbf{x}_l, \mathbf{y}_l)$, $a + b = 1$, $k = ak_s + bk_l$, $F' = \sum_{i \in \mathcal{F}} f_i y'_i = aF_s + bF_l$, $C' = \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} c_{ij} x'_{ij} = aC_s + bC_l$, $\alpha' = a\alpha'_s + b\alpha'_l$, $\beta' = a\beta'_s + b\beta'_l$, $z' = az'_s + bz'_l$ and $\theta' = \theta_2$. Moreover, $(\alpha', \beta', z', \theta')$ is a feasible dual solution to the *KFTRA* problem.

Proof. From the constructions of $(\alpha'_l, \beta'_l, z'_l)$ and $(\alpha'_s, \beta'_s, z'_s)$, we get $\forall i \in \mathcal{F} : \sum_{j \in \mathcal{C}} \beta'_{s,ij} \leq f_i + z'_{s,i} + \theta_2$ and $\sum_{j \in \mathcal{C}} \beta'_{l,ij} \leq f_i + z'_{l,i} + \theta_1$, then $\sum_{j \in \mathcal{C}} \beta'_{ij} \leq f_i + z'_i + a\theta_2 + b\theta_1 \leq f_i + z'_i + \theta_2$ after multiplying the first inequality by a , the second by b and adding them together. In addition, with the setting $\forall i \in \mathcal{F}, j \in \mathcal{C} : \beta'_{ij} = \alpha'_j - c_{ij}$, we get the feasibility of $(\alpha', \beta', z', \theta')$ to LP (6). Next, we aim to derive the following bound

$$\left(2 + \frac{1}{N}\right) F_l + C_l \leq \left(2 + \frac{1}{N}\right) \left(\sum_{j \in \mathcal{C}} r_j \alpha'_{l,j} - \sum_{i \in \mathcal{F}} R_i z'_{l,i} - k_l \theta_2 \right) \quad (12)$$

from the inequality (10). For now, suppose this bound holds, from (11), we have

$$\left(2 + \frac{1}{N}\right) F_s + C_s \leq \left(2 + \frac{1}{N}\right) \left(\sum_{j \in \mathcal{C}} r_j \alpha'_{s,j} - \sum_{i \in \mathcal{F}} R_i z'_{s,i} - k_s \theta_2 \right) \quad (13)$$

. After multiplying (12) by b , (13) by a and adding them together, we get

$$\left(2 + \frac{1}{N}\right) F' + C' \leq \left(2 + \frac{1}{N}\right) \left(\sum_{j \in \mathcal{C}} r_j \alpha'_j - \sum_{i \in \mathcal{F}} R_i z'_i - k \theta_2 \right) \quad (14)$$

. This then yields the lemma together with the feasibility of $(\boldsymbol{\alpha}', \boldsymbol{\beta}', \mathbf{z}', \theta')$ and $\text{cost}(\mathbf{x}', \mathbf{y}') = F' + C'$. The last thing left is to verify in the following that (12) indeed holds from the inequality (10), the termination condition of the algorithm $\theta_2 - \theta_1 \leq \epsilon = \frac{c_{\min}}{8N^2}$ and the fact that $C_l = \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} c_{ij} x_{l,ij} \geq c_{\min}$.

$$\begin{aligned} C_l &\leq 2 \left(\sum_{j \in \mathcal{C}} r_j \alpha'_{l,j} - \sum_{i \in \mathcal{F}} R_i z'_{l,i} - k_l \theta_1 - F_l \right) \\ &\leq 2 \left(\sum_{j \in \mathcal{C}} r_j \alpha'_{l,j} - \sum_{i \in \mathcal{F}} R_i z'_{l,i} - k_l \theta_2 - F_l \right) + \frac{c_{\min} k_l}{4N^2} \\ &\leq 2 \left(\sum_{j \in \mathcal{C}} r_j \alpha'_{l,j} - \sum_{i \in \mathcal{F}} R_i z'_{l,i} - k_l \theta_2 - F_l \right) + \frac{C_l k_l}{4N^2} \end{aligned}$$

For simplicity, let $\Delta = \left(\sum_{j \in \mathcal{C}} r_j \alpha'_{l,j} - \sum_{i \in \mathcal{F}} R_i z'_{l,i} - k_l \theta_2 - F_l \right)$. Because $k_l \leq N$ and $N \geq 1$, we get $C_l \leq \frac{2}{1 - \frac{k_l}{4N^2}} \Delta \leq \left(2 + \frac{1}{N}\right) \Delta$. Hence, the inequality (12) is verified.

For runtime, our BS procedure totally makes $O(L + \log N + \log n)$ probes (L is the number of bits of the input costs) over the interval $[0, \frac{n c_{\max}}{2}]$ until the interval becomes the size of $\frac{c_{\min}}{8N^2}$. Moreover, each probe takes $O(n^4)$ to invoke the SPD algorithm, so the total time is $O(n^4 (L + \log N + \log n))$ which dominates the overall runtime of the algorithm PK.

For the next step, we focus on analyzing the GP and RR procedures to bound $\text{cost}(\mathbf{x}, \mathbf{y}) = F + C$ with $\text{cost}(\mathbf{x}', \mathbf{y}') = F' + C'$. Note that F and C are the expected total facility and connection costs respectively from the randomized procedure RR. This procedure can also be derandomized using the method of conditional expectation as in [14] for the k -median problem. In the following, we bound F with F' and C with C' separately. With probability 1, RR opens exactly k facilities. Specifically, it randomly opens each facility in \mathbf{y}_p with probability b , and each facility in $\bar{\mathbf{y}}_p$ with probability $\frac{k - k_s}{k_l - k_s}$ which is also b . Since GP properly splits the vector \mathbf{y}_l into \mathbf{y}_p and $\bar{\mathbf{y}}_p$ s.t. $\mathbf{y}_l = \mathbf{y}_p + \bar{\mathbf{y}}_p$, we can conclude that

each facility in \mathbf{y}_l is opened with probability b . In addition, RR randomly opens each facility in \mathbf{y}_s with probability a , therefore the total expected opening cost is $aF_s + bF_l$ which is F' .

Lemma 17. *The total expected facility opening cost F satisfies $F = F'$.*

Next, we bound C with C' . Suppose two $FTRA$ instances with the solutions $(\mathbf{x}_s, \mathbf{y}_s)$ and $(\mathbf{x}_l, \mathbf{y}_l)$ are produced from the BS procedure. Afterwards, for getting a feasible solution to $KFTRA$ from these solutions, instead we consider a naive pseudo-polynomial time algorithm. The algorithm first treats the $FTRA$ instances with the solutions $(\mathbf{x}_s, \mathbf{y}_s)$ and $(\mathbf{x}_l, \mathbf{y}_l)$ as equivalent $FTFL$ instances (by naively splitting sites and keeping the clients unchanged) with the transformed solutions $(\check{\mathbf{x}}_s, \check{\mathbf{y}}_s)$ and $(\check{\mathbf{x}}_l, \check{\mathbf{y}}_l)$ ⁴ respectively. Then, it uses the matching and rounding procedures (cf. the paragraph before Lemma 7.1 in [25]) on $(\check{\mathbf{x}}_s, \check{\mathbf{y}}_s)$ and $(\check{\mathbf{x}}_l, \check{\mathbf{y}}_l)$ to get a feasible solution $(\check{\mathbf{x}}, \check{\mathbf{y}})$ to $KFTFL$. Finally, the solution $(\check{\mathbf{x}}, \check{\mathbf{y}})$ can be easily transformed to a feasible solution (\mathbf{x}, \mathbf{y}) to $KFTRA$. Now, the important observation is that directly applying GP and RR to these $FTRA$ instances with the solutions $(\mathbf{x}_s, \mathbf{y}_s)$ and $(\mathbf{x}_l, \mathbf{y}_l)$ from BS obtains essentially the *same solution* (\mathbf{x}, \mathbf{y}) (also the same cost) to $KFTRA$ as the naive algorithm does. It is mainly because for the $FTRA$ instances of size $O(n)$, our designed GP procedure pairs the integer vectors in polynomial time. This is the speed-up of the matching procedure therein [25] applied to the equivalent $FTFL$ instances of size $O(\sum_{i \in \mathcal{F}} R_i)$. Therefore, *only in the analysis*, we can consider the naive algorithm instead to get the following bound for C . This analysis trick is similar to the trick used for analyzing the algorithm SGA (cf. the paragraph before Theorem 5).

Lemma 18. *The total expected connection cost C satisfies $C \leq (1 + \max(a, b)) C'$.*

Proof. For the equivalent $FTFL$ instances, we let \mathcal{F}' be the set of split facilities with size $\sum_{i \in \mathcal{F}} R_i$ and use k to index these facilities. After the matching and rounding procedures in [25] on the transformed solutions $(\check{\mathbf{x}}_s, \check{\mathbf{y}}_s)$ and $(\check{\mathbf{x}}_l, \check{\mathbf{y}}_l)$, we get the solution $(\check{\mathbf{x}}, \check{\mathbf{y}})$ to $KFTFL$. Also from its Lemma 7.2, we can directly obtain the bound $C_j \leq (1 + \max(a, b)) \sum_{k \in \mathcal{F}'} c_{kj} (ax_{s,kj} + bx_{l,kj})$ ⁵ where $C_j = \sum_{k \in \mathcal{F}'} c_{kj} x_{kj}$, i.e. the expected connection cost of any client j . Since $(\check{\mathbf{x}}_s, \check{\mathbf{y}}_s)$, $(\check{\mathbf{x}}_l, \check{\mathbf{y}}_l)$ and (\mathbf{x}, \mathbf{y}) are transformed from $(\mathbf{x}_s, \mathbf{y}_s)$, $(\mathbf{x}_l, \mathbf{y}_l)$ and $(\check{\mathbf{x}}, \check{\mathbf{y}})$ respectively with the same costs, we have

⁴ W.l.o.g., the solutions can be easily transformed between $FTRA$ and $FTFL$ as shown in Theorem 7 of [18].

⁵ Note that we get this tighter bound coefficient $(1 + \max(a, b))$ (rather than 2) through a bit more careful analysis of the lemma therein.

$$\begin{aligned}
C &= \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} c_{ij} x_{ij} = \sum_{j \in \mathcal{C}} \sum_{k \in \mathcal{F}'} c_{kj} \check{x}_{kj} \\
&\leq (1 + \max(a, b)) \sum_{j \in \mathcal{C}} \sum_{k \in \mathcal{F}'} c_{kj} (a x_{s, kj} + b x_{l, kj}) \\
&= (1 + \max(a, b)) \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} c_{ij} (a x_{s, ij} + b x_{l, ij}) \\
&= (1 + \max(a, b)) C'
\end{aligned}$$

, which concludes the lemma.

Adding up the separate bounds in the previous two lemmas, we get $\text{cost}(\mathbf{x}, \mathbf{y}) = F + C \leq F' + (1 + \max(a, b)) C'$. Relating this bound to the bound $(2 + \frac{1}{N}) F' + C' \leq (2 + \frac{1}{N}) \text{cost}(\boldsymbol{\alpha}', \boldsymbol{\beta}', \mathbf{z}', \theta')$ in Lemma 16, we obtain

$$\begin{aligned}
\text{cost}(\mathbf{x}, \mathbf{y}) &\leq F' + (1 + \max(a, b)) C' \\
&< \left(2 + \frac{1}{N}\right) (1 + \max(a, b)) F' + (1 + \max(a, b)) C' \\
&\leq \left(2 + \frac{1}{N}\right) (1 + \max(a, b)) \text{cost}(\boldsymbol{\alpha}', \boldsymbol{\beta}', \mathbf{z}', \theta') \\
&< 4 \text{cost}(\boldsymbol{\alpha}', \boldsymbol{\beta}', \mathbf{z}', \theta')
\end{aligned}$$

. The last inequality is from the fact that $a = \frac{k_l - k}{k_l - k_s} \leq 1 - \frac{1}{N}$ (achieved when $k_l = N$ and $k_s = k - 1$), and $b = \frac{k - k_s}{k_l - k_s} \leq 1 - \frac{1}{k}$ (achieved when $k_l = k + 1$ and $k_s = 1$). Therefore, $1 + \max(a, b) \leq 2 - \frac{1}{N}$ and $(2 + \frac{1}{N})(1 + \max(a, b)) \leq 4 - \frac{1}{N^2}$. By the weak duality theorem, the approximation ratio is 4. For runtime, from the algorithm PK, both GP and RR take $O(n^2)$.

Theorem 6. *Algorithm PK is 4-approximation for the uniform KFTRA in polynomial time $O(n^4(L + \log N + \log n))$.*

7 Concluding Remarks

In this paper, we studied the Constrained Fault-Tolerant Resource Allocation (*FTRA*) problem and its important variant Constrained Fault-Tolerant k -Resource Allocation (*KFTRA*) problem. In particular, although the Fault-Tolerant Facility Location (*FTFL*) problem is a special case of *FTRA*, we have shown that it is not much harder (in terms of the attained approximation ratios) to approximate *FTRA* than *FTFL* for both the general and the uniform cases. The counterparts of *FTRA* is the Unconstrained Fault-Tolerant Resource Allocation (*FTRA* _{∞}) problem. This problem was recently claimed in [29] to be easier to approximate than *FTFL* and slightly harder than the classical Uncapacitated Facility Location (*UFL*) problem.

From the practical side, our developed resource allocation models inherited from *FTFL* and *UFL* are more general and applicable than these classical models. Therefore, in future, it is worth looking at these models' other important variants such as the capacitated variant in [18], the Reliable Resource Allocation (*RRA*) problem in [19] and etc. From the theoretical side, two grand challenges still remain today: 1) close the approximation gap between *FTFL* (1.7245) and *UFL* (1.488) or show *FTFL* is more difficult than *UFL*; 2) reduce the ratio of 1.488 to the established lower bound 1.463 for *UFL*.

Acknowledgment

We would like to thank Li Yan, Marek Chrobak and Jaroslaw Byrka for their helpful discussions.

Funding

This research was partially supported by Australian Research Council Discovery Project grant #DP0985063, National Science Foundation of China under its General Projects funding #61170232 and Research Initiative Grant of Beijing Jiaotong University #2010RC018.

References

1. Jaroslaw Byrka and Karen Aardal. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *SIAM Journal on Computing*, 39(6):2212–2231, 2010.
2. Jaroslaw Byrka, MohammadReza Ghodsi, and Aravind Srinivasan. Lp-rounding algorithms for facility-location problems. Technical report, 2012. <http://arxiv.org/abs/1007.3611>, CoRR.
3. Jaroslaw Byrka, Aravind Srinivasan, and Chaitanya Swamy. Fault-tolerant facility location: A randomized dependent lp-rounding algorithm. In *Proceedings of Integer Programming and Combinatorial Optimization, 14th International Conference (IPCO)*, pages 244–257, Lausanne, Switzerland, 9–11 June 2010. Springer-Verlag, Berlin.
4. Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for facility location problems. *SIAM J. Comput.*, 34(4):803–824, 2005.
5. Fabián A. Chudak and David B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM J. Comput.*, 33(1):1–25, 2003.
6. Toshihiro Fujito and Hidekazu Kurahashi. A better-than-greedy algorithm for k-set multicover. In *Proceedings of the Third international conference on Approximation and Online Algorithms (WAOA)*, pages 176–189, Palma de Mallorca, Spain, 6–7 October 2005. Springer-Verlag, Berlin.
7. Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31:228–248(21), April 1999.
8. Sudipto Guha, Adam Meyerson, and Kamesh Munagala. A constant factor approximation algorithm for the fault-tolerant facility location problem. *J. Algorithms*, 48(2):429–440, 2003.

9. Dorit S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22(1):148–162, 1982.
10. Q.S. Hua, D. Yu, F.C. Lau, and Y. Wang. Exact algorithms for set multicover and multiset multicover problems. In *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC)*, pages 34–44, Honolulu, Hawaii, USA, 16–18 December 2009. Springer-Verlag, Berlin.
11. Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50(6):795–824, 2003.
12. Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing (STOC)*, pages 731–740, Montreal, Quebec, Canada, 19–21 May 2002. ACM, New York, NY.
13. Kamal Jain and Vijay V. Vazirani. An approximation algorithm for the fault tolerant metric facility location problem. In *Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 177–183, Saarbrücken, Germany, 5–8 September 2000. Springer-Verlag, Berlin.
14. Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k -median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, 2001.
15. S.G. Kolliopoulos. Approximating covering integer programs with multiplicity constraints. *Discrete applied mathematics*, 129(2):461–473, 2003.
16. Bernhard Korte and Jens Vygen. *Combinatorial Optimization*. Springer-Verlag, Berlin, Third Edition, 2006.
17. Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. In *Proceedings of ICALP*, pages 77–88, Zurich, Switzerland, 4–8 July 2011. Springer-Verlag, Berlin.
18. Kewen Liao and Hong Shen. Unconstrained and constrained fault-tolerant resource allocation. In *Proceedings of the 17th annual international conference on computing and combinatorics (COCOON)*, pages 555–566, Dallas, Texas, USA, 14–16 August 2011. Springer-Verlag, Berlin.
19. Kewen Liao and Hong Shen. Approximating the reliable resource allocation problem using inverse dual fitting. In *Proceedings of the Eighteenth Computing: The Australasian Theory Symposium (CATS)*, page Vol. 128, Melbourne, Australia, January–February 2012. ACS, Sydney.
20. Mohammad Mahdian. *Facility Location and the Analysis of Algorithms through Factor-Revealing Programs*. PhD thesis, MIT, USA, 2004.
21. Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay Vazirani. A greedy facility location algorithm analyzed using dual fitting. In *Proceedings of the 4th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 127–137, Berkeley, CA, USA, 18–20 August 2001. Springer-Verlag, Berlin.
22. Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Approximation algorithms for metric facility location problems. *SIAM J. Comput.*, 36(2):411–432, 2006.
23. David B. Shmoys, Eva Tardos, and Karen Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 265–274, El Paso, Texas, USA, 4–6 May 1997. ACM, New York.

24. Maxim Sviridenko. An improved approximation algorithm for the metric uncapacitated facility location problem. In *Proceedings of the 9th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 240–257, Cambridge, MA, USA, 27-29 May 2002. Springer-Verlag, Berlin.
25. Chaitanya Swamy and David B. Shmoys. Fault-tolerant facility location. *ACM Trans. Algorithms*, 4(4):1–27, 2008.
26. Shihong Xu and Hong Shen. The fault-tolerant facility allocation problem. In *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC)*, pages 689–698, Honolulu, Hawaii, USA, December 16-18 2009. Springer-Verlag, Berlin.
27. Li Yan and Marek Chrobak. Approximation algorithms for the fault-tolerant facility placement problem. *Information Processing Letters*, 111(11):545 – 549, 2011.
28. Li Yan and Marek Chrobak. New results on the fault-tolerant facility placement problem. Technical report, 2011. <http://arxiv.org/abs/1108.5471>, CoRR.
29. Li Yan and Marek Chrobak. Lp-rounding algorithms for the fault-tolerant facility placement problem. Technical report, 2012. <http://arxiv.org/abs/1205.1281>, CoRR.